Table of Contents

Pivotal

PCF v2.0 Feature Highlights

This topic highlights important new features included in Pivotal Cloud Foundry (PCF) v2.0.

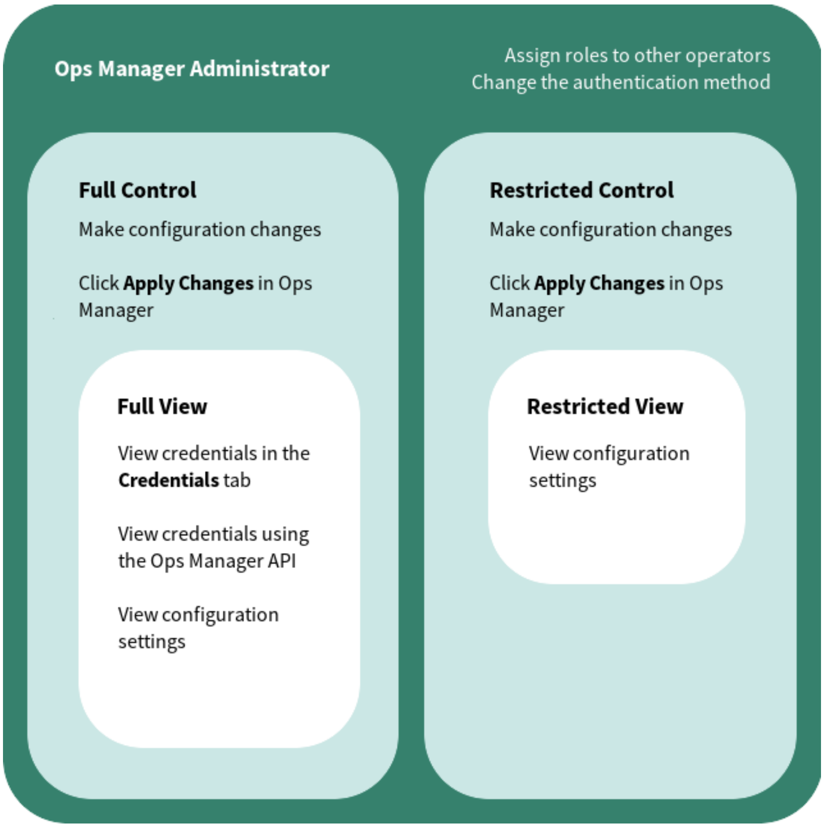
Note: Elastic Runtime has been renamed Pivotal Application Service.

Ops Manager Highlights

Ops Manager v2.0 includes the following major features:

RBAC Support

Ops Manager v2.0 introduces support for role-based access control (RBAC).



You can assign the following roles to determine which operators in your organization make deployment changes, view credentials, and manage user roles in Ops Manager:

- **Ops Manager Administrator**: Full control over all Ops Manager configuration and settings, including assigning user roles and managing authentication
- **Full Control**: Write access to Ops Manager, including credentials
- **Restricted Control**: Write access to Ops Manager with no access to credentials
- **Full View**: Read-only access to Ops Manager UI and API endpoints, including credentials
- **Restricted View**: Read-only access to Ops Manager UI and API endpoints with no access to credentials

See Configuring Role-Based Access Control (RBAC) in Ops Manager for more information.

Custom Login Banner

In Ops Manager v2.0, operators can set a custom banner that every user sees when they log in to the Ops Manager BOSH Director. To set the banner, edit the **Custom SSH Banner** field in the **Director Config** page of the Ops Manager tile.

See the Configuring Ops Manager topic for your IaaS for more information.

- Amazon Web Services
- Google Cloud Platform
- Microsoft Azure
- OpenStack
- vSphere

Azure Stack Support (Beta)

Operators can deploy Ops Manager v2.0 to Microsoft Azure in their own local datacenter using Azure Stack.

See Deploying BOSH and Ops Manager to Azure Manually for more information.

BOSH System Metrics Available in Loggregator Firehose

PCF now forwards BOSH health metrics generated for all VMs in a deployment to the Loggregator Firehose by default. For more information about this feature and its implementation, see the *BOSH System Metrics Forwarder* section in the Overview of the Loggregator System.

BOSH CLI Renamed

Similar to previous Ops Manager versions, the Ops Manager VM includes both versions of the BOSH CLI. In Ops Manager v2.0, both versions of the BOSH CLI have been renamed.

If you used BOSH CLI v2 in earlier versions of Ops Manager, you ran commands using `bosh2`. In Ops Manager v2.0, run the same commands using `bosh`. For example, see the following table to compare the changes to the `bosh vms` command:

BOSH CLI Version	PCF v1.12	PCF v2.0
BOSH CLI v1	`bosh vms`	`bosh-old vms`
BOSH CLI v2	`bosh2 -e MY-ENV vms`	`bosh -e MY-ENV vms`

Many BOSH CLI v1 commands are incompatible with the BOSH Director. Pivotal recommends using BOSH CLI v2 commands for compatibility with future versions of PCF.

VMware NSX-T Networking Support

Ops Manager v2.0 adds support for VMware NSX-T networking. NSX is a networking solution for VMware that provides a firewall, load balancing, and NAT/SNAT services for PCF. NSX-T is intended to work across multiple clouds and provide networking for container platforms. Previous versions of Ops Manager supported NSX-V.

When you upgrade from a previous version of Ops Manager with NSX networking enabled, Ops Manager defaults to NSX-V. The NSX-T integration is only for fresh installs of PCF. You can enable NSX-T networking by selecting **NSX-T** in the new **NSX Mode** dropdown menu of the **vCenter Config** pane.

See Configuring Ops Manager on vSphere for more information.

Operators can additionally use the NSX Manager to configure policies for PCF applications. See the NSX-T Container Plug-in for Kubernetes and Cloud Foundry - Installation and Administration Guide for more information.

Note: You must have NSX-T v2.1 installed to use this integration.

Note: The IPSec add-on is not supported with NSX-T.

Breaking Change: If you opt out of the BOSH DNS feature, your PCF deployment cannot support NSX-T networking.

Pivotal Application Service (PAS) Highlights

Pivotal Application Service (PAS) is the new name for Elastic Runtime.

VMware NSX-T Networking Support

PAS v2.0 adds support for VMware NSX-T networking. NSX is a networking solution for VMware that provides a firewall, load balancing, and NAT/SNAT services for PCF. NSX-T is intended to work across multiple clouds and provide networking for container platforms. Previous versions of PAS supported NSX-V networking.

To use NSX-T networking, you must install the NSX-T tile.

Warning: The NSX-T integration is only for fresh installs of PCF. You cannot upgrade an existing deployment to use NSX-T, and there is no upgrade path from NSX-V to NSX-T.

To enable NSX-T networking for your PCF installation, you must perform the following steps:

1. In the Ops Manager Director tile > **vCenter Config** pane, select **NSX-T** from the **NSX Mode** drop-down menu. See Step 2: vCenter Config Page in *Configuring Ops Manager on vSphere* for more information.

2. Install the NSX-T tile.
   - You must install the NSX-T tile after you install the Ops Manager Director tile.
   - You must install the NSX-T tile before you install the PAS tile.

3. In the PAS tile > **Networking** pane, under **Container Network Plugin Interface**, select **External**.

Operators can additionally use the NSX Manager to configure policies for PCF applications. See the NSX-T Container Plug-in for Kubernetes and Cloud Foundry - Installation and Administration Guide for more information.

Note: You must have NSX-T v2.1 installed to use this integration.

Note: The IPSec add-on is not supported with NSX-T.

Breaking Change: If you opt out of the BOSH DNS feature, your PCF deployment cannot support NSX-T networking.

Secure Service Instance Credentials

The PAS tile now includes its own CredHub VM to support the secure storage of service instance credentials. Previously, PCF could only use the Cloud Controller database for storing these credentials.

Release-Level Backup and Restore

PAS v2.0 includes support for release-level backup and restore. BOSH Backup and Restore (BBR) now backs up each PAS component using scripts specific to the component. This new flow ensures services stop using the component database before it is backed up and improves the correctness of the backup. The steps to backup and restore with BBR are unchanged. For more information about component and service availability during backup, see PAS Component Availability During Backup

Colocated Errands on Instance VMs

The PAS tile no longer includes individual errand VMs. Instead, the errand jobs are colocated on other VMs in the deployment. Colocated errands run faster than traditional errands and use fewer resources, including disk and IP space. These errands are now set to always run by default.

Apps Manager Highlights

Option to Enable Metrics Forwarder on Apps Manager

Apps Manager includes a new option to bind the Metrics Forwarder service to an application. To bind the Metrics Forwarder service to an application, open **Services** and click *Add a Service*.

Option to Enable PCF Scheduler on Apps Manager

Apps Manager includes a new option to bind the PCF Scheduler service to an application. To bind the PCF Scheduler service to an application, open **Services** and click *Add a Service*.

Option to Enable App Autoscaler on Apps Manager

Apps Manager includes a new option to bind the App Autoscaler service to an application. To bind the App Autoscaler service to an application, open **Services** and click *Add a Service*. To access the configuration panel for App Autoscaler, click **Manage**.

Mappings Endpoint Integrated in Apps Manager

Apps Manager now integrates the Spring Boot Actuator `/mappings` endpoint. This endpoint displays the endpoints an app serves and other related details. For more information, see Using Actuators.

PCF Isolation Segment Highlights

VMware NSX-T Networking Support

PCF Isolation Segment v2.0 adds support for VMware NSX-T networking. NSX is a networking solution for VMware that provides a firewall, load balancing, and NAT/SNAT services for PCF. NSX-T is intended to work across multiple clouds and provide networking for container platforms. Previous versions of PCF Isolation Segment supported NSX-V networking.

To use NSX-T networking, you must install the NSX-T tile.

Warning: The NSX-T integration is only for fresh installs of PCF. You cannot upgrade an existing deployment to use NSX-T, and there is no upgrade path from NSX-V to NSX-T.

To enable NSX-T networking for your PCF installation, you must perform the following steps:

1. In the Ops Manager Director tile > **vCenter Config** pane, select **NSX-T** from the **NSX Mode** drop-down menu. See Step 2: vCenter Config Page in *Configuring Ops Manager on vSphere* for more information.

2. Install the NSX-T tile.
   - You must install the NSX-T tile after you install or upgrade the Ops Manager Director tile.
   - You must install the NSX-T tile before you install or upgrade the PAS tile.

3. In the PCF Isolation Segment tile > **Networking** pane, under **Container Network Plugin Interface**, select **External**.

Operators can additionally use the NSX Manager to configure policies for PCF applications. See the NSX-T Container Plug-in for Kubernetes and Cloud Foundry - Installation and Administration Guide for more information.

Note: You must have NSX-T v2.1 installed to use this integration.

Note: The IPSec add-on is not supported with NSX-T.

Breaking Change: If you opt out of the BOSH DNS feature, your PCF deployment cannot support NSX-T networking.

Gorouter and HAProxy Support Multiple Certificates

You can now add more than one certificate for Gorouter and HAProxy in the **Networking** configuration pane. This improves security and removes the need to reissue the existing certificate when you want to add TLS support for custom domains. Gorouter and HAProxy use SNI to determine the correct certificate to present in a TLS handshake. For more information, see the Multiple Certificates section of *Securing Traffic into Cloud Foundry*.

Services Highlights

Spring Cloud Services v1.5

Configurable data services allow you more control over where Spring Cloud Services (SCS) data resides. SCS now allows you to configure a service name and plan for creating a service broker MySQL database and RabbitMQ queues.

Configured server backend enhancements add support for `pattern` and `searchPaths` to Git backend properties.

The SCS service broker stores service instance credentials in CredHub. Client applications with updated SCS Connectors automatically resolve credentials stored in CredHub.

Spring Cloud Edgware builds on Spring Boot v1.5.x.

MySQL for PCF v2.2

MySQL for PCF v2.2 now offers the ability to create leader-follower, multi-AZ instances. Operators can monitor leader-follower instances with replication metrics.

Developers can configure MySQL for read/write/mixed workloads.

Developers can customize MySQL usernames in service bindings and service keys.

Developers and operators can create read-only access credentials.

RabbitMQ for PCF v1.11

RabbitMQ for PCF v1.11 is compatible with PCF v2.0.

Five configurable on-demand plans in the RabbitMQ tile enable operators to create customized plans to match developer demands.

Redis for PCF v1.11

Redis for PCF v1.11 is compatible with PCF v2.0 while taking advantage of the newest BOSH and Ops Manager features.

The Redis for PCF tile now uses SHA2 checksums for all releases.

Users that require backups can use the Redis for PCF On-Demand service.

Pivotal

Pivotal Cloud Foundry Release Notes

Pivotal Cloud Foundry is certified by the Cloud Foundry Foundation for 2017.

Read more about the certified provider program and the requirements of providers.

This topic provides links to the release notes for Pivotal Cloud Foundry (PCF) and PCF services. Release notes include new features, breaking changes, bug fixes, and known issues.

PCF Release Notes

- PCF v2.0 Breaking Changes
- Pivotal Application Service (PAS) Release Notes
- PCF Ops Manager Release Notes
- PAS for Windows 2012R2 Release Notes
- PCF Isolation Segment Release Notes
- Stemcell Release Notes
- Windows Stemcell Release Notes
- BOSH Backup and Restore Release Notes

PCF Services Release Notes

- App Distribution for PCF
- GemFire for PCF
- MySQL for PCF
- Application Watchdog for PCF (Beta)
- PCF JMX Bridge
- PCF Log Search
- PCF Metrics
- PCF Service Broker for AWS
- Push Notification Service for PCF
- Rabbit MQ for PCF
- Redis for PCF
- Session State Caching Powered by GemFire
- Single Sign-On for PCF
- Spring Cloud Services on PCF
- Scheduler for PCF (Beta)

PCF v2.0 Breaking Changes

This topic describes the breaking changes you need to be aware of when upgrading to Pivotal Cloud Foundry (PCF) v2.0. For more information about important preparation steps you must follow before beginning an upgrade, see Upgrading Pivotal Cloud Foundry.

Pivotal Application Service (PAS)

SSO Tile Requires Upgrade

If you have the Single Sign-On (SSO) tile installed and you are upgrading to PCF v2.0, you must upgrade the SSO tile to v1.5.3 and configure the Apps Manager errand before installing PCF v2.0. To properly prepare for upgrading to PCF 2.0 with the SSO tile, see Upgrade SSO Service Tile resource instance between PCF 1.12 and PCF 2.0 in the Pivotal Knowledge Base.

Internal Pivotal Application Service Credentials

Several internal credentials that PAS uses for inter-component communication are now generated and stored in BOSH CredHub instead of Ops Manager. If you want to access these credentials, you must use the CredHub CLI or the Ops Manager API instead of accessing the **Credentials** tab of the PAS tile.

For a list of the credentials migrated to BOSH CredHub, see the*Migration of Internal Credentials to BOSH CredHub* section of the Pivotal Application Service v2.0 Release Notes topic.

If you opt out of the BOSH DNS feature, your PCF deployment cannot support Secure Service Instance Credentials feature.

Upgrading from Earlier Versions of Elastic Runtime

You must upgrade first to a version of Elastic Runtime v1.12 to successfully upgrade to PAS v2.0.

If you are upgrading from a PCF deployment that at one point included Elastic Runtime v1.7.16 or earlier, you must perform the remedial steps outlined in App Usage Data and Events Data Become Corrupted After Upgrade or Install before proceeding with this upgrade. If you fail to perform the remedial steps for this issue, this upgrade process may corrupt your existing usage data.

Configuring Router Behavior for Client Certificates while Deploying PAS

For some deployment configurations, you must configure how PCF handles x-forwarded-client-cert (XFCC) HTTP headers based on where TLS is terminated for the first time in your deployment. In the **Router behavior for Client Certificates field**, you cannot select the **Router does not request client certificates option**.

Ops Manager

BOSH CLI Renamed

Similar to previous Ops Manager versions, the Ops Manager virtual machine includes both versions of the BOSH CLI. In Ops Manager v2.0, both versions of the BOSH CLI have been renamed.

If you used BOSH CLI v2 in earlier versions of Ops Manager, you ran commands using `bosh2`. In Ops Manager v2.0.0, run the same commands using `bosh`. For example, see the following table to compare the changes to the `bosh vms` command:

BOSH CLI Version	PCF v1.12	PCF v2.0
BOSH CLI v1	`bosh vms`	`bosh-old vms`
BOSH CLI v2	`bosh2 -e MY-ENV vms`	`bosh -e MY-ENV vms`

Many BOSH CLI v1 commands are incompatible with the BOSH Director. Pivotal recommends using BOSH CLI v2 commands for compatibility with future versions of PCF.

Disabling BOSH DNS for Ops Manager Director

Do not disable BOSH DNS without consulting Pivotal Support.

For more information about disabling BOSH DNS, see Disabling or Opting Out of BOSH DNS in PCF in the Pivotal Knowledge Base.

`current_record` Property

The `current_record` property is now reserved. You can no longer create a new property named `current_record` in a collection record.

PCF Isolation Segment

Configuring Router Behavior for Client Certificates While Installing PCF Isolation Segment

For some deployment configurations, you must configure how PCF handles x-forwarded-client-cert (XFCC) HTTP headers based on where TLS is terminated for the first time in your deployment. In the **Router behavior for Client Certificates field**, you cannot select the **Router does not request client certificates option**.

Pivotal

Pivotal Application Service v2.0 Release Notes

Pivotal Cloud Foundry is certified by the Cloud Foundry Foundation for 2017.

Read more about the certified provider program and the requirements of providers.

Note: Elastic Runtime has been renamed Pivotal Application Service.

Releases

2.0.0

Component	Version
Stemcell	3468.13
backup-and-restore-sdk	1.2.1
binary-offline-buildpack	1.0.15
bosh-dns-aliases	0.0.1
bosh-system-metrics-forwarder	0.0.11
capi	1.44.0*
cf-autoscaling	104
cf-backup-and-restore	0.0.10
cf-mysql	36.10.0
cf-networking	1.8.1
cf-smoke-tests	39
cf-syslog-drain	2
cflinuxfs2	1.175.0
consul	187
credhub	1.6.5
diego	1.29.0
dotnet-core-offline-buildpack	1.0.30
garden-runc	1.9.4
go-offline-buildpack	1.8.13
grootfs	0.30.0
haproxy	8.4.2
java-offline-buildpack	4.6
loggregator	99*
mysql-backup	1.37.0
mysql-monitoring	8.12.0
nats	22
nfs-volume	1.1.3
nodejs-offline-buildpack	1.6.10
notifications	44
notifications-ui	31
php-offline-buildpack	4.3.43
pivotal-account	1.8.2
push-apps-manager-release	663.0.4
push-usage-service-release	664.0.9

Components marked with an asterisk have been patched to resolve security vulnerabilities or fix component behavior.

Component	Version
python-offline-buildpack	1.6.1
routing	0.168.0
ruby-offline-buildpack	1.7.5
service-backup	18.1.5
staticfile-offline-buildpack	1.4.18
statsd-injector	1.0.30
syslog-migration	10
uaa	52.4
** Components marked with an asterisk have been patched to resolve security vulnerabilities or fix component behavior.*	

How to Upgrade

The procedure for upgrading to Pivotal Application Service (PAS) v2.0 is documented in the [Upgrading Pivotal Cloud Foundry](#) topic.

When upgrading to v2.0, be aware of the following upgrade considerations:

- To successfully upgrade to PAS v2.0, you must first upgrade to a version of Elastic Runtime v1.12.
- Some partner service tiles may be incompatible with PCF v2.0. Pivotal is working with partners to ensure their tiles are updated to work with the latest versions of PCF.

  For information about which partner service releases are currently compatible with PCF v2.0, review the appropriate partners services release documentation at https://docs.pivotal.io, or contact the partner organization that produces the tile.

New Features in PAS v2.0

BOSH DNS Service Discovery for Application Containers (Beta)

In PCF v2.0, application containers look up services using the BOSH DNS service discovery mechanism.

To support this lookup, BOSH Director colocates a BOSH DNS server on every deployed VM. This colocation is a prerequisite for migrating completely to BOSH DNS in a future release of PCF. However, this colocation does not impact the current behavior of DNS for Cloud Foundry components in PCF v2.0. System components still use consul to discover and locate other Cloud Foundry components.

You can opt out of deploying BOSH DNS in PCF v2.0. For more information, see the Ops Manager v2.0 Release Notes and Disabling or Opting Out of BOSH DNS in PCF in the Pivotal Knowledge Base.

BOSH System Metrics Available in Loggregator Firehose

PCF now forwards BOSH health metrics generated for all VMs in a deployment to the Loggregator Firehose by default. For more information about this feature and its implementation, see the *BOSH System Metrics Forwarder* section in the Overview of the Loggregator System.

The new flow of BOSH system metrics cannot be disabled. Therefore, if you are currently using the PCF JMX Bridge tile or the BOSH HM Forwarder to consume them, you may receive duplicate data. To prevent this, you can do the following:

- Stop using PCF JMX Bridge to consume BOSH system metrics outside of the Firehose. See Preventing Duplicated BOSH System Metrics in PCF JMX Bridge for additional instructions.
- Uninstall the BOSH HM Forwarder.

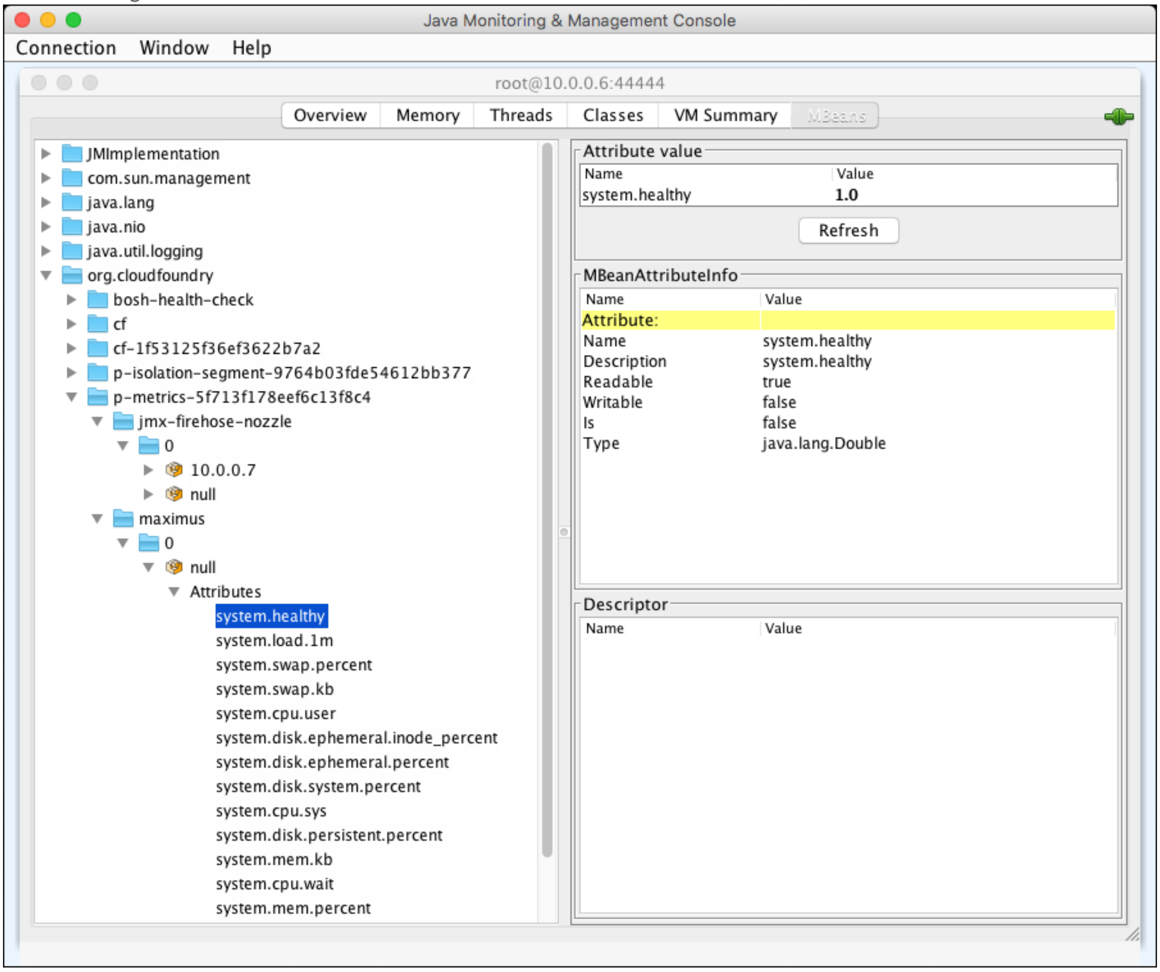
For guidance information about configuring the Ops Manager Director, see the Ops Manager configuration topic for your IaaS:

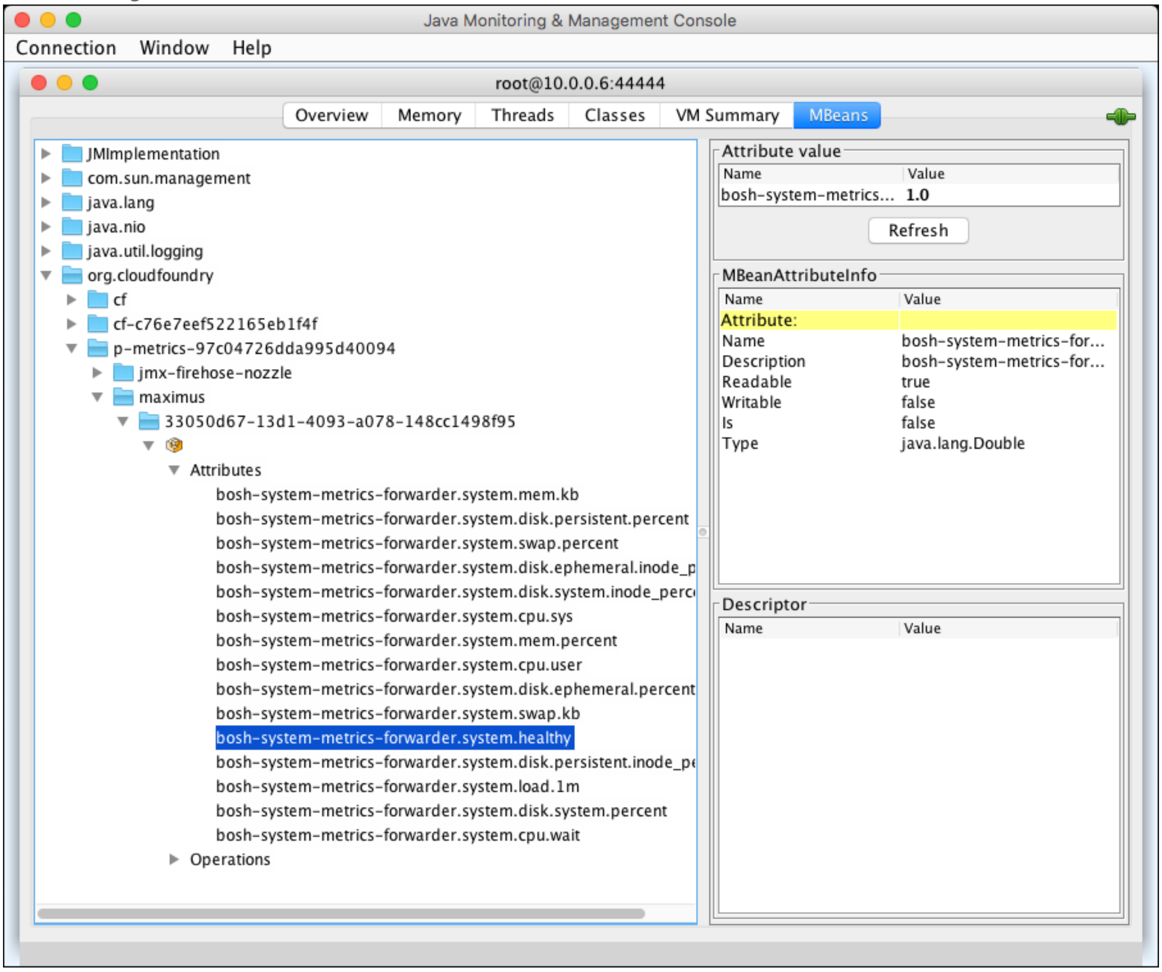
- Configuring Ops Manager Director on AWS (Terraform)
- Manually Configuring Ops Manager Director for AWS
- Configuring Ops Manager on Azure
- Configuring Ops Manager Director on GCP

- [Configuring Ops Manager Director for OpenStack]
- [Configuring Ops Manager on vSphere]

Metric Namespace Difference in JMX Bridge

Because BOSH System metrics now come from the Firehose, their namespaces are different in PCF JMX Bridge. For an explanation of how metric names differ between PCF 2.0 and earlier versions, see the following table.

PCF Version	Explanation
1.12 and earlier	**Example Metric:** `system.healthy` **Description:** The BOSH Director delivers the metric name. The metric is nested in the tree structure by deployment name, VM name, VM instance number, and attributes for that VM instance. The sub-node of VM instance number is always named null. **Reference Image:** 

2.0	**Example Metric:** `bosh-system-metrics-forwarder.system.healthy` **Description:** The Firehose delivers the metric name. The tree shows the VM GUID instead of the VM instance number and the sub-node is always empty. This namespacing affects all previous BOSH health metrics. **Reference Image:** 

Gorouter and HAProxy Trust the Diego Instance Identity Intermediate CA

The trust between the Gorouter and HAProxy enables mutual authentication between applications that run on PCF. The Gorouter and HAProxy are configured with the root certificate authority. This occurs automatically within PCF.

Gorouter and HAProxy Trust Additional CAs

When validating client requests using mutual TLS, the Gorouter trusts multiple certificate authorities (CAs) by default. Operators can now configure the Gorouter and HAProxy to trust custom CAs in addition to well-known, public CAs and Ops Manager Director Trusted Certificates.

For more information about configuring this feature, see the PAS installation instructions for your IaaS:

- Deploying PAS on AWS (Terraform)
- Manually Configuring PAS for AWS
- Deploying PAS on Azure
- Deploying PAS on GCP (Terraform)
- Installing PAS after Deploying PCF on OpenStack
- Configuring PAS for vSphere

Gorouter and HAProxy Support Multiple Certificates

You can now add more than one certificate for Gorouter and HAProxy in the **Networking** configuration pane. This improves security and removes the need to reissue the existing certificate when you want to add TLS support for custom domains. Gorouter and HAProxy use SNI to determine the correct certificate to present in a TLS handshake. For more information, see the Multiple Certificates section of *Securing Traffic into Cloud Foundry*.

XFCC Support for Deployments that Terminate TLS at HAProxy

PCF now supports XFCC header configuration for deployments that terminate TLS for the first time at HAProxy. In addition, the selection options for this configuration field have been renamed to reflect differences in XFCC configuration based on TLS termination entry points. For more information, see the PAS installation instructions for your IaaS:

- Deploying PAS on AWS (Terraform)
- Manually Configuring PAS for AWS
- Deploying PAS on Azure
- Deploying PAS on GCP (Terraform)
- Installing PAS after Deploying PCF on OpenStack
- Configuring PAS for vSphere

Migration of Internal Credentials to BOSH CredHub

Several internal credentials, the `secret` and `simple_credentials` that PAS uses for inter-component communication, are now generated and stored in BOSH CredHub instead of the Ops Manager database.

This is part of an ongoing effort to migrate all credentials to BOSH CredHub, which will reduce the amount of places credentials are stored, aid in credential rotation, and increase security. For information about the internal credentials that were migrated to BOSH CredHub in 1.12, see the Migration of Internal Credentials to BOSH CredHub section of the *Pivotal Elastic Runtime v1.12 Release Notes* topic.

To access the following credentials, you must use the CredHub CLI or the Ops Manager API instead of accessing the **Credentials** tab of the PAS tile. For instructions about retrieving PAS credentials, see Retrieving Credentials from Your Deployment.

- `.autoscaling.broker_credentials`
- `.autoscaling.encryption_key`
- `.backup-prepare.backup_encryption_key`
- `.diego_database.bbs_encryption_passphrase`
- `.nfs_server.blobstore_secret`
- `.notifications.encryption_key`
- `.push-pivotal-account.encryption_key`
- `.push-usage-service.secret_token`
- `.router.route_services_secret`
- `.properties.consul_encrypt_key`
- `.nats.credentials`

VMware NSX-T Networking Support

PAS v2.0 adds support for VMware NSX-T networking. NSX is a networking solution for VMware that provides a firewall, load balancing, and NAT/SNAT services for PCF. NSX-T is intended to work across multiple clouds and provide networking for container platforms. Previous versions of PAS supported NSX-V networking.

To use NSX-T networking, you must install the NSX-T tile.

Warning: The NSX-T integration is only for fresh installs of PCF. You cannot upgrade an existing deployment to use NSX-T, and there is no upgrade path from NSX-V to NSX-T.

To enable NSX-T networking for your PCF installation, you must perform the following steps:

1. In the Ops Manager Director tile > **vCenter Config** pane, select **NSX-T** from the **NSX Mode** drop-down menu. See Step 2: vCenter Config Page in *Configuring Ops Manager on vSphere* for more information.
2. Install the NSX-T tile.
   - You must install the NSX-T tile after you install the Ops Manager Director tile.
   - You must install the NSX-T tile before you install the PAS tile.
3. In the PAS tile > **Networking** pane, under **Container Network Plugin Interface**, select **External**.

Operators can additionally use the NSX Manager to configure policies for PCF applications. See the NSX-T Container Plug-in for Kubernetes and Cloud Foundry - Installation and Administration Guide for more information.

Note: You must have NSX-T v2.1 installed to use this integration.

Note: The IPSec add-on is not supported with NSX-T.

Breaking Change: If you opt out of the BOSH DNS feature, your PCF deployment cannot support NSX-T networking.

Secure Service Instance Credentials (Beta)

The PAS tile now includes its own CredHub VM to support the secure storage of service instance credentials. Previously, PCF could only use the Cloud Controller database for storing these credentials.

Enabling this feature requires the following:

- PCF operators must follow the steps in Securely Storing Service Instance Credentials with PAS CredHub.
- PCF service authors must modify their service to store instance credentials in PAS CredHub. Alongside the PAS v2.0 release, the Spring Cloud Services tile is the first service to support storing its instance credentials in PAS CredHub.

For more information about CredHub in PCF, see the CredHub Documentation.

Breaking Change: If you opt out of the BOSH DNS feature, your PCF deployment cannot support Secure Service Instance Credentials feature.

Release-Level Backup and Restore

PAS v2.0 includes support for release-level backup and restore. BOSH Backup and Restore (BBR) now backs up each PAS component using scripts specific to the component. This new flow ensures services stop using the component database before it is backed up and improves the correctness of the backup. The steps to backup and restore with BBR are unchanged. For more information about component and service availability during backup, see PAS Component Availability During Backup

Colocated Errands on Instance VMs

The PAS tile no longer includes individual errand VMs. Instead, the errand jobs are colocated on other VMs in the deployment. Colocated errands run faster than traditional errands and use fewer resources, including disk and IP space. These errands are now set to always run by default.

Context Objects for Service Bindings

A context object is sent to service brokers when provisioning a service instance containing platform-specific information, such as the organization and space GUIDs. PAS v2.0 adds the same context object to binding requests. If the service broker wants to make decisions based on the organization or space GUIDs in which the binding is created, it can do so.

For more information, see Open Service Broker API.

Syslog Scheduler Highly Available During Zone Outage

The Syslog Scheduler is now scalable. The number of Syslog Scheduler instances defaults to `2`. This is the minimum number of instances necessary to make the syslog drain highly available.

Option to Enable Metrics Forwarder on Apps Manager

Apps Manager includes a new option to bind the Metrics Forwarder service to an application. To do so, go to **Services** and click **Add a Service**.

Option to Enable PCF Scheduler on Apps Manager

Apps Manager includes a new option to bind the PCF Scheduler service to an application. To do so, go to **Services** and click **Add a Service**.

Option to Enable App Autoscaler on Apps Manager

Apps Manager includes a new option to bind the App Autoscaler service to an application. To do so, go to **Services** and click **Add a Service**. To access the configuration panel for App Autoscaler, click **Manage**.

Apps Manager Shows Multiple Buildpacks

In Apps Manager, the app page **Settings** tab shows multiple buildpacks for apps pushed with multiple buildpacks.

Mappings Endpoint Integrated in Apps Manager

Apps Manager now integrates the Spring Boot Actuator `/mappings` endpoint. This endpoint displays the endpoints an app serves and other related details. For more information, see Using Actuators.

Terraform Templates for Installing PCF

The PAS v2.0 release on Pivotal Network includes Terraform template downloads. Terraform is a tool for creating and updating infrastructure resources that helps provide a better and more consistent PCF install experience on multiple IaaS providers. PCF internal development teams maintain and test Terraform templates in many scenarios.

This release includes templates for GCP and AWS. If you are installing PCF on either of these IaaS providers, you can use the Terraform templates to automatically create the necessary infrastructure resources. For instructions, see the following topics:

- Installing PCF on GCP using Terraform.
- Installing PCF on AWS using Terraform.

> **Note**: The AWS Terraform template replaces the PCF Cloudformation for AWS Setup file on Pivotal Network.

Option to Enable Self-Service Network Policies

Operators can now enable self-service network policies for Space Developers. Prior to this release, enabling access occurred on either a per-developer or per-group basis. For more information about enabling this feature, see the Installing Pivotal Cloud Foundry topic for your IaaS.

Known Issues

Tasks Unreliable

There is a bug in the Cloud Controller that affects the operation of tasks. Any time a user runs a task, the following may happen:

- The task cancels before completing.
- The task is reported as a failure even if it finishes successfully.

Preventing Duplicated BOSH System Metrics in PCF JMX Bridge

PCF now forwards BOSH health metrics generated for all VMs in a deployment to the Loggregator Firehose by default. See BOSH System Metrics Available in Loggregator Firehose for more information.

The new flow of BOSH system metrics cannot be disabled. Therefore, if you are currently using the PCF JMX Bridge tile to consume them, you may receive duplicate data. To prevent this, delete **JMX Provider IP Address** in **Director Config** of your Ops Manager Director tile.

Deleting the IP address means that BOSH system metrics will no longer be sent to JMX Bridge using the direct connection from the BOSH Director to the JMX Provider. As these BOSH system metrics are now available in JMX Bridge by default through its Firehose nozzle, breaking the prior direct connection by deleting the JMX Provider IP address prevents the duplication of BOSH metrics for JMX Bridge consumers.

About Advanced Features

The Advanced Features section of the PAS tile includes new functionality that may have certain constraints.

Although these features are fully supported, Pivotal recommends caution when using them in production.

Pivotal

PCF Ops Manager Release v2.0 Release Notes

Pivotal Cloud Foundry is certified by the Cloud Foundry Foundation for 2017.

Read more about the certified provider program and the requirements of providers.

How to Upgrade

2.0.0

Ops Manager v2.0.0 uses the following component versions:

Component	Version
Stemcell	3468.11
BOSH Director	264.3
CredHub	1.6.5
UAA	52.4
AWS CPI	67
Azure CPI	29
GCP CPI	25.10.0
OpenStack CPI	34
vSphere CPI	45
** Components marked with an asterisk have been patched to resolve security vulnerabilities or fix component behavior.*	

New Features in Ops Manager v2.0

Role-Based Access Control

Ops Manager v2.0 introduces support for role-based access control (RBAC). You can assign the following roles to determine which operators in your organization make deployment changes, view credentials, and manage user roles in Ops Manager:

- **Ops Manager Administrator**
- **Full Control**
- **Restricted Control**
- **Full View**
- **Restricted View**

See Configuring Role-Based Access Control (RBAC) in Ops Manager for more information.

Custom Director Login Banner

In Ops Manager v2.0, operators can set a custom banner that every user sees when they log in to the BOSH Director. To set the banner, edit the **Custom SSH Banner** field in the **Director Config** page of the Ops Manager tile.

See the Director configuration topic for your IaaS for more information. For example, see Configuring Ops Manager on vSphere.

Azure Stack Support (Beta)

Operators can deploy Ops Manager v2.0 to Microsoft Azure in their own local datacenter using Azure Stack.

See Launching an Ops Manager Director Instance on Azure without an ARM Template for more information.

BOSH CLI Renamed

Similar to previous Ops Manager versions, the Ops Manager VM includes both versions of the BOSH CLI. In Ops Manager v2.0, both versions of the BOSH CLI have been renamed.

If you used BOSH CLI v2 in earlier versions of Ops Manager, you ran commands using `bosh2`. In Ops Manager v2.0, run the same commands using `bosh`. For example, see the following table to compare the changes to the `bosh vms` command:

BOSH CLI Version	PCF v1.12	PCF v2.0
BOSH CLI v1	`bosh vms`	`bosh-old vms`
BOSH CLI v2	`bosh2 -e MY-ENV vms`	`bosh -e MY-ENV vms`

Many BOSH CLI v1 commands are incompatible with the BOSH Director. Pivotal recommends using BOSH CLI v2 commands for compatibility with future versions of PCF.

BOSH NATS Traffic Uses TLS

The BOSH Director communicates with the agents in your deployment over NATS. For added security, Ops Manager v2.0 sends all NATS traffic using Transport Layer Security (TLS) encryption.

See Component: Messaging (NATS) for more information about NATS.

BOSH DNS Service Discovery (Beta) and Opt-Out Option

In PCF v2.0, application containers look up services using the BOSH DNS service discovery mechanism. To support this lookup, BOSH Director colocates a BOSH DNS server on every deployed VM.

Since the BOSH DNS feature is beta in PCF v2.0, Pivotal recommends that you deploy or upgrade your PCF v2.0 deployment first on a non-production environment.

You can opt out of deploying BOSH DNS servers on every VM by selecting the **Disable BOSH DNS server for troubleshooting purposes** option in Operations Manager. For more information on this configuration option, see the *Configuring Ops Manager Director* instructions for your IaaS. For example, if you are deploying PCF on GCP, see Configuring Ops Manager Director on GCP.

For more information about disabling BOSH DNS, see Disabling or Opting Out of BOSH DNS in PCF in the Pivotal Knowledge Base.

BOSH System Metrics Server Colocated on BOSH Director

Ops Manager v2.0 colocates the new BOSH Metrics Server on the BOSH Director and includes a UAA client with the correct authorities and scopes. This colocation allows BOSH system metrics to flow into the Loggregator system by default.

For more information about this feature, see the *BOSH System Metrics Forwarder* section in the Overview of the Loggregator System topic and BOSH System Metrics Available in Loggregator Firehose in the PAS Release Notes.

VMware NSX-T Networking Support

Ops Manager v2.0 adds support for VMware NSX-T networking. NSX is a networking solution for VMware that provides a firewall, load balancing, and NAT/SNAT services for PCF. NSX-T is intended to work across multiple clouds and provide networking for container platforms. Previous versions of Ops Manager supported NSX-V.

When you upgrade from a previous version of Ops Manager with NSX networking enabled, Ops Manager defaults to NSX-V. The NSX-T integration is only for fresh installs of PCF. You can enable NSX-T networking by selecting **NSX-T** in the new **NSX Mode** dropdown menu of the **vCenter Config** pane.

See Configuring Ops Manager on vSphere for more information.

Operators can additionally use the NSX Manager to configure policies for PCF applications. See the NSX-T Container Plug-in for Kubernetes and Cloud

Foundry - Installation and Administration Guide for more information.

Note: You must have NSX-T v2.1 installed to use this integration.

Note: The IPSec add-on is not supported with NSX-T.

Breaking Change: If you opt out of the BOSH DNS feature, your PCF deployment cannot support NSX-T networking.

Ops Manager Minimum Disk Size Warning

In Ops Manager v2.0, the web interface displays a warning banner if the Ops Manager appliance VM disk is less than 50 GB in size. If this warning appears, resize your Ops Manager VM disk to 50 GB or larger.

See the Ops Manager installation topic for your IaaS for more information:

- Configuring Ops Manager on AWS Deployed Manually
- Deploying BOSH and Ops Manager to Azure Manually
- Deploying BOSH and Ops Manager to GCP
- Provisioning the OpenStack Infrastructure
- Deploying BOSH and Ops Manager to vSphere

Configure Colocated Errands

Tile authors can configure the errands defined in their product tile to run on existing VMs in a deployment. Colocated errands run faster than traditional errands and use fewer resources, including disk and IP space.

See Tile Errands in the PCF Tile Developers Guide for more information.

Configure Tiles with Runtime Configs

Tile authors can include `runtime_configs` as a top-level key in tile metadata to define global deployment configurations. Named runtime config settings apply to all VMs in a deployment.

Version v2.0 of Ops Manager supports defining any number of `runtime_configs` in an existing tile. Tile authors can also create a tile that includes a runtime config only and does not define any job types or errands.

See Managing Runtime Configs in the PCF Tile Developers Guide for more information.

Named Manifests

Ops Manager v2.0 supports specifying and rendering named manifests in a collection property. For more information, see the named_manifest section of the Product Template Reference topic.

Bug Fixes

- **[Bug Fix]** Ops Manager v2.0 fixes a bug where the UI showed a confusing name for a collection record starting with a certificate.
- **[Bug Fix]** Ops Manager v2.0 fixes a bug where the stemcell hardening work unexpectedly blocked non-root access to ping Ops Manager, resulting in a verification error.
- **[Bug Fix]** Ops Manager v2.0 fixes a bug where changes could not be applied until BOSH Director was deployed.
- **[Bug Fix]** Ops Manager v2.0 fixes a bug where the dashboard took a long time to load.
- **[Bug Fix]** Ops Manager v2.0 fixes a bug where BOSH Director failed to create certifications because the BOSH agent certification was created for the wrong network.

PAS for Windows 2012R2 v2.0 Release Notes

Note: Elastic Runtime has been renamed Pivotal Application Service.

Releases

2.0.0

Component	Version
Stemcell	1200.7
consul	187
diego	1.29.0
event_log	0.2
garden-windows	0.12.0
hwc-offline-buildpack	2.3.11
loggregator	99*
windows-utilities	0.4.0
** Components marked with an asterisk have been patched to resolve security vulnerabilities or fix component behavior.*	

How to Upgrade

The Pivotal Application Service (PAS) for Windows 2012R2 v2.0 tile is available with the release of Pivotal Cloud Foundry (PCF) version v2.0. To use the PAS for Windows 2012R2 tile, you need Ops Manager v2.0 or later and PAS v2.0 or later.

New Features in PAS for Windows 2012R2 v2.0

RDP Option Now Available in VM Options

The **VM Options** pane in the **PAS for Windows** tile now optionally supports connection to all virtual machines (VMs) through Remote Desktop Protocol (RDP).

KMS Configuration Option Now Available in VM Options

The **VM Options** pane in the **PAS for Window 2012R2** tile now allows you to enable and configure KMS with a hostname and port. Previously, this had to be performed manually by editing the runtime config. For more information, see Configuring a KMS Host.

Colocated Errands on Instance VMs

The PAS tile no longer includes individual errand VMs. Instead, the errand jobs are colocated on other VMs in the deployment. Colocated errands run faster than traditional errands and use fewer resources, including disk and IP address space. These errands are now set to always run by default.

About Advanced Features

The **Advanced Features** pane of the PAS for Windows 2012R2 tile includes new functionality that may have certain constraints. Although these features are fully supported, Pivotal recommends caution when using them in production.

Known Issues

KMS Activation Does Not Work

The firewall rule that is created to allow outbound KMS activation does not set the correct destination port.

PCF Isolation Segment v2.0 Release Notes

Releases

Component	Version
Stemcell	3468.13
cf-networking	1.8.1
cflinuxfs2	1.175.0
consul	187
diego	1.29.0
garden-runc	1.9.4
grootfs	0.30.0
haproxy	8.4.2
loggregator	99*
nfs-volume	1.1.3
routing	0.168.0
syslog-migration	10
** Components marked with an asterisk have been patched to resolve security vulnerabilities or fix component behavior.*	

About PCF Isolation Segment

The PCF Isolation Segment v2.0 tile is available for installation with PCF v2.0.

How to Install

To install a PCF Isolation Segment, you must first install PCF v2.0.

New Features in PCF Isolation Segment v2.0

Gorouter and HAProxy Trust the Diego Instance Identity Intermediate CA

The trust between the Gorouter and HAProxy enables mutual authentication between applications that are running on PCF. The Gorouter and HAProxy are configured with the root certificate authority. This occurs automatically within PCF.

Gorouter and HAProxy Trust Additional CAs

When validating client requests using mutual TLS, the Gorouter trusts multiple certificate authorities (CAs) by default. Operators can now configure the Gorouter and HAProxy to trust custom CAs in addition to well-known, public CAs and Ops Manager Director Trusted Certificates. For more information about configuring this feature, see the Installing PCF Isolation Segment topic.

Gorouter and HAProxy Support Multiple Certificates

You can now add more than one certificate for Gorouter and HAProxy in the **Networking** configuration pane. This improves security and removes the need to reissue the existing certificate when you want to add TLS support for custom domains. Gorouter and HAProxy use SNI to determine the correct certificate to present in a TLS handshake. For more information, see the Multiple Certificates section of *Securing Traffic into Cloud Foundry*.

XFCC Support for Deployments that Terminate TLS at HAProxy

PCF now supports XFCC header configuration for deployments that terminate TLS for the first time at HAProxy. In addition, the selection options for this configuration field have been renamed to reflect differences in XFCC configuration based on TLS termination entry points. For more information about configuring this feature, see the Installing PCF Isolation Segment topic.

VMware NSX-T Networking Support

PCF Isolation Segment v2.0 adds support for VMware NSX-T networking. NSX is a networking solution for VMware that provides a firewall, load balancing, and NAT/SNAT services for PCF. NSX-T is intended to work across multiple clouds and provide networking for container platforms. Previous versions of PCF Isolation Segment supported NSX-V networking.

To use NSX-T networking, you must install the NSX-T tile.

Warning: The NSX-T integration is only for fresh installs of PCF. You cannot upgrade an existing deployment to use NSX-T, and there is no upgrade path from NSX-V to NSX-T.

To enable NSX-T networking for your PCF installation, you must perform the following steps:

1. In the Ops Manager Director tile > **vCenter Config** pane, select **NSX-T** from the **NSX Mode** drop-down menu. See Step 2: vCenter Config Page in *Configuring Ops Manager on vSphere* for more information.

2. Install the NSX-T tile.
   - You must install the NSX-T tile after you install or upgrade the Ops Manager Director tile.
   - You must install the NSX-T tile before you install or upgrade the PAS tile.

3. In the PCF Isolation Segment tile > **Networking** pane, under **Container Network Plugin Interface**, select **External**.

Operators can additionally use the NSX Manager to configure policies for PCF applications. See the NSX-T Container Plug-in for Kubernetes and Cloud Foundry - Installation and Administration Guide for more information.

Note: You must have NSX-T v2.1 installed to use this integration.

Note: The IPSec add-on is not supported with NSX-T.

Breaking Change: If you opt out of the BOSH DNS feature, your PCF deployment cannot support NSX-T networking.

About Advanced Features

The Advanced Features section of the PCF Isolation Segment tile includes new functionality that may have certain constraints. Although these features are fully supported, Pivotal recommends caution when using them in production.

Stemcell Release Notes

This topic includes release notes for stemcells used with Pivotal Cloud Foundry (PCF) versions 2.0.x.

3468.13

Release Date: December 8, 2017

- Bump Ubuntu Trusty stemcell USN-3509-2: Linux kernel (Xenial HWE) vulnerabilities

Windows Stemcell Release Notes

This topic includes release notes for Windows stemcells used with Pivotal Application Service (PAS) for Windows 2012R2.

To download a stemcell, see the Stemcells for PCF (Windows) on Pivotal Network.

1200.11

Release Date: December 22, 2017

- For Azure, GCP & AWS Windows Stemcells, the root disk (C Drive) will be automatically resized on creation to the disk size specified in BOSH cloud config. Due to current CPI limitations, vSphere Stemcells are NOT able to resize their root disk on creation.
- Intended for use with December Microsoft security updates.

1200.10

Release Date: December 19, 2017

- You must use stembuild version 0.13 when creating a 1200.10 stemcell by hand.
- AWS stemcells repartition to use entire root disk size as specified in BOSH cloud config.
- Stemcell adds support for multiple CPIs. You can now set `stemcell_formats` in stemcell.MF.
- Intended for use with November Microsoft security updates.
- Updated OpenSSH to 0.0.22.
- The BOSH Agent uses a lock file to ensure that DNS resolvers are updated only on first startup.

1200.8

Release Date: November 10, 2017

- BOSH Agent: Disables port 5985 for WinRM by default.
- [Bug Fix] Fixes an issue where an empty cloud config would remove all DNS resolvers from a Windows host.
- [Bug Fix] Fix for IPsec add-on.

Known Issues

- File `updates.txt` is not generated for 2016/1709 stemcells.

1200.7

Release Date: October 23, 2017

- [Bug Fix] BOSH Agent timeout fix for high ESX workload scenarios.
- Intended for 2017 Oct Windows Updates roll-up (KB4041685).

1200.6

Release Date: October 18, 2017

- [Security Improvement] Includes CIS MS-L1 v2.2.1 security hardening.
- [Security Improvement] The security policies disable RDP by default. To enable RDP, use the `enable_rdp` job in the windows-utilities-release (version 0.4.0 or greater).

- [Bug Fix] Fixes an issue in the BOSH Agent regarding DNS resolvers that can cause application downtime when a BOSH Director is unavailable (e.g. during upgrades) when deployed on Cloud Foundry.

Known Issues

- In the case of an empty cloud config, the Windows host DNS list will be cleared on BOSH Agent restarts.
- CIS policies break the IPsec add-on.

1200.5

Release Date: October 11, 2017

- Install-CFFeatures is now Install-CFFeatures2012.
- [Security Improvement] BOSH Agent randomizes password for Administrator user on bootup. To set the password, use the `set_password` job in the windows-utilities-release.
- Removes Windows Defender for all IaaSes in Windows Server 2016/1709.
- [Improvement] No longer installs Docker on Windows 2016/1709.

1200.4

Release Date: September 14, 2017

- The BOSH-Agent now disables automatic updates during its bootstrap process.
- Do not remove Powershell-ISE when building stemcell.
- Added better error checking when applying group policies.
- Intended for 2017 Sep Windows Updates roll-up.
- Sets smaller MTU of network interfaces created by Docker on GCP for Windows 2016.
- Skip sysprep until official Windows 1709 build is available due to bug in insider build.

1200.3

Release Date: August 22, 2017

- Agent backs off exponentially when unable to reach the director, moving from 5 seconds to 160 seconds over 6 connection attempts, to reduce the impact on small-footprint BOSH VMs. This resolves BOSH Agent Open Issue #137.
- BOSH SSH is now supported as a beta feature. Users can enable connecting to a cmd session using the `bosh ssh` command by running the relevant job from windows-utilities-release.
- Fixed an issue where jobs were being stopped synchronously rather than concurrently, preventing stop scripts that waited on other stop scripts from ever finishing.
- Fixed an issue where jobs that failed to start on the first attempt weren't being retried.
- Other minor bug fixes and performance improvements.

1200.0

Release Date: July 14, 2017

- Includes July 2017 Windows Security Updates.
- Fixes an error where Windows stemcells were incompatible with bosh director setting 'enable_nats_delivered_templates' set to true.
- Fixed startup issue on GCP.
- Fixed issue where the Windows Agent would reset DNS settings whenever the HTTPMetadataService was invoked on AWS.
- Upgrades included .NET version to 4.7.

1079.0

Release Date: June 5, 2017

- Based on Windows Server 2012R2.
- Includes .NET Framework 4.6.1.
- Available for AWS, GCP, and Azure.
- Includes all Windows Updates and security patches up through April 2017.
- To be used with Pivotal Cloud Foundry (PCF) Runtime for Windows v1.9.3+, v1.10.2+, and v1.11.0.

1056.1

Release Date: June 1, 2017

- Based on Windows Server 2012R2.
- Includes .NET Framework 4.6.1.
- Available for AWS, GCP, and Azure.
- Includes all Windows Updates and security patches up through March 2017.
- To be used with Pivotal Cloud Foundry (PCF) Runtime for Windows v1.9.0, v1.9.1, v1.9.2, v1.10.0, v1.10.1.

1056.0

Release Date: April 5, 2017

- Based on Windows Server 2012R2.
- Includes .NET Framework 4.6.1.
- Available for AWS, GCP, and Azure.
- Includes all Windows Updates and security patches up through March 2017.
- To be used with Pivotal Cloud Foundry (PCF) Runtime for Windows v1.9.0, v1.9.1, v1.9.2, v1.10.0, v1.10.1.

Known Limitations

- Does not support BOSH SSH or persistent disks.

BOSH Backup and Restore Release Notes

This topic includes release notes for the versions of BOSH Backup and Restore (BBR) used with Pivotal Cloud Foundry (PCF) versions 1.12.x.

Releases

v1.1.3

Release Date: October 13, 2017

Features

- Output order of locking

Bug Fixes

None

Known Issues

None

v1.1.2

Release Date: October 12, 2017

Features

- Upgrade to golang 1.9

Bug Fixes

None

Known Issues

None

v1.1.1

Release Date: October 2, 2017

Features

- Adds improved logging

Bug Fixes

None

Known Issues

None

v1.1.0

Release Date: September 26, 2017

Features

- Adds “restore-cleanup”, which can be run after a failed restore to remove artifacts from, and unlock, the deployments being restored.
- Adds support for BOSH Deployments and Directors to lock deployments during the restore process, and to declare the order that locking is done.

Bug Fixes

None

Known Issues

None

v1.0

Release Date: August 7, 2017

Compatibility

Compatible with PCF v1.12

In PCF 1.12, Ops Manager export/import no longer includes releases. Customers must back up and restore their BOSH Director using BBR to ensure releases are backed up and restored. Manual backup and restore steps are deprecated in PCF 1.12

Features

BBR v1.0 enables backup and restore of BOSH Deployments and BOSH Directors. BBR uses backup and restore functionality that is packaged in the BOSH Deployment or Director. This functionality is present in PCF v1.11 and later.

Bug Fixes

None

Known Issues

None

Architecture and Installation Overview

For PCF Architects and Operators

This guide shows you how to design a Pivotal Cloud Foundry (PCF) platform and install it on an IaaS. If you are doing this, you have one or both of the following roles:

- **Architects** design a PCF platform. They know the IaaS that they will deploy it to, and what other relevant resources they have. In their design, they consider needs for the platform's capacity, availability, security, geography, budget, and other factors. If they do not install PCF themselves, they provide the architectural specifications to whoever does.
- **Operators** run a PCF platform, keep it up-to-date, monitor its health and performance, and fix any problems. They may also install the platform, or perform "Day 2" configurations that expand its functionality and integrate it with external systems.

This guide helps people in both roles create a PCF platform that does what they want it to. The contents of this guide follow the phases of a typical PCF planning and installation effort.

> **Note:** Elastic Runtime has been renamed Pivotal Application Service.

Planning and Installation Overview

PCF is a suite of products that runs on multiple IaaSes. Planning and installing PCF means building layers from the bottom up, starting with the details of your IaaS and ending with "Day 2" configurations that you perform on a installed and running PCF deployment.

Here's the typical PCF planning and installation process:

1. **Plan**
   - Review the Requirements for your IaaS (AWS, Azure, GCP, OpenStack, vSphere).
   - Refer to the Reference Architecture for your IaaS.
   - Assess your platform needs, including capacity, availability, container support, host OS, resource isolation, and geographical distribution. Discuss with your Pivotal contact.
2. **Deploy BOSH and Ops Manager**
   - BOSH is an open-source tool that lets you run software systems in the cloud.
     - BOSH and its IaaS-specific Cloud Provider Interfaces (CPIs) are what enable PCF to run on multiple IaaSes.
     - See Deploying with BOSH for a brief description of the BOSH deployment process.
   - Ops Manager is a GUI application deployed by BOSH that streamlines deploying subsequent software to the cloud via BOSH.
     - Ops Manager represents PCF products as *tiles* with multiple configuration panes that let you input or select configuration values needed for the product.
     - Ops Manager generates BOSH manifests containing the user-supplied configuration values, and sends them to the Director.
     - After you install Ops Manager and BOSH, you use Ops Manager to deploy almost all PCF products.
   - Deploying Ops Manager deploys both BOSH and Ops Manager with a single procedure.
     - On AWS, you can deploy Ops Manager manually, or automatically with a CloudFormation template.
     - On Azure, you can deploy Ops Manager manually, or automatically with an Azure Resource Manager (ARM) template. On Azure Government Cloud and Azure Germany you can only deploy Ops Manager manually.
3. **Deploy BOSH Add-ons** (Optional)
   - BOSH add-ons include the IPsec, ClamAV, and File Integrity Monitoring, which enhance PCF platform security and security logging.
   - You deploy these add-ons via BOSH rather than installing them with Ops Manager tiles.
4. **Install Runtimes**
   - PAS (Pivotal Application Service) lets developers develop and manage cloud-native apps and software services.
     - PAS is based on the Cloud Foundry Foundation's open-source Application Runtime (formerly Elastic Runtime) project.
   - PCF Isolation Segment lets a single PAS deployment run apps from separate, isolated pools of computing, routing, and logging resources.
     - Operators replicate and configure an Isolation Segment tile for each new resource pool they want to create.
     - You must install PAS before you can install Isolation Segment.

- PAS for Windows 2012R2 enables PAS to manage Windows Server 2012 cells hosting .NET apps, and can also be replicated to create multiple isolated resource pools.
    - As with Isolation Segment, Operators replicate and configure a PAS for Windows 2012R2 tile for each new resource pool they want to create.
    - You must install PAS before you can install PAS for Windows 2012R2.
- Small Footprint PAS is an alternative to PAS that uses far fewer VMs than PAS but has limitations.

5. **Day 2 Configurations**
    - Day 2 configurations set up internal operations and external integrations on a running PCF platform.
        - Examples include front-end configuration, user accounts, logging and monitoring, internal security, and container and stemcell images.
6. **Install Services**
    - Install software services for PCF developers to use in their apps.
        - Services include the databases, caches, and message brokers that stateless cloud apps rely on to save information.
        - Installing and managing software services on PCF is an ongoing process, and is covered in the PCF Operator Guide.

Guide Contents

This guide has two parts. The first part explains the PCF planning and installation process, and the second describes the main tools that operators use when installing PCF.

Planning and Installation

PCF is a suite of products that runs on multiple IaaSes. Planning and installing PCF means building layers from the bottom up, starting with the details of your IaaS and ending with "Day 2" configurations that you perform on a installed and running PCF deployment.

The first part of this guide follows this bottom-up progression:

- Planning PCF - Design a PCF platform that runs on your IaaS and fits your needs.
- Deploying BOSH and Ops Manager - Build the foundation of PCF.
- Deploying BOSH Add-Ons - Use BOSH to extend foundation-level PCF capabilities.
- Installing Runtimes - Install the application and container runtime environments that PCF exists for.
- Installing Platform Extension Tiles - Use Ops Manager to extend PCF platform capabilities.
- "Day 2" Configurations - Set up internal operations and external integrations for PCF.

After installing PCF, Operators install the software services that PCF developers use in their apps. These PCF services include the databases, caches, and message brokers that stateless cloud apps rely on to save information.

Installing and managing software services on PCF is an ongoing process, and is covered in the PCF Operator Guide.

Tools and Reference

The second part of this guide explains the main PCF tools that operators use when installing PCF:

- Using Ops Manager
- Using the Cloud Foundry Command Line Interface (cf CLI)
- Troubleshooting and Diagnostics

Related Documentation

The PCF Operator Guide explains how to maintain a running PCF platform, including monitoring, tuning, troubleshooting, and upgrading.

The PCF Security Guide explains how PCF security sytems work and how to keep your PCF platform secure.

Getting Started with PCF gives a high-level overview of how PCF works and explains how you can try a a simple deployment on your own local machine.

See Pivotal Cloud Foundry Documentation for all PCF documentation.

Preparing Your Firewall

Page last updated:

This topic describes how to configure your firewall for Pivotal Cloud Foundry (PCF) and how to verify that PCF resolves DNS entries behind your firewall.

Configure Your Firewall for PCF

Ops Manager and Pivotal Application Service (PAS) require the following open TCP ports:

- **25555**: Routes from Ops Manager to the Ops Manager Director.
- **443**: Routes to HAProxy or, if configured, your own load balancer
- **80**: Routes to HAProxy or, if configured, your own load balancer
- **2222**: Necessary in order to connect using Application SSH. For details, see the Diego Architecture Diagram

UDP port **123** must be open if you want to use an external NTP server.

For more information about required ports for additional installed products, refer to the Network Communication Paths and other product documentation.

The following example procedure uses iptables commands to configure a firewall.

> **Note:** `GATEWAY_EXTERNAL_IP` is a placeholder. Replace this value with your `PUBLIC_IP`.

1. Open `/etc/sysctl.conf`, a file that contains configurations for Linux kernel settings, with the command below:

```
$ sudo vi /etc/sysctl.conf
```

2. Add the line `net.ipv4.ip_forward=1` to `/etc/sysctl.conf` and save the file.

3. If you want to remove all existing filtering or Network Address Translation (NAT) rules, run the following commands:

```
$ iptables --flush
$ iptables --flush -t nat
```

4. Add environment variables to use when creating the IP rules:

```
$ export INTERNAL_NETWORK_RANGE=10.0.0.0/8
$ export GATEWAY_INTERNAL_IP=10.0.0.1
$ export GATEWAY_EXTERNAL_IP=203.0.113.242
$ export PIVOTALCF_IP=10.0.0.2
$ export HA_PROXY_IP=10.0.0.254
```

5. Run the following commands to configure IP rules for the specified chains:
   - **FORWARD**:

```
$ iptables -A FORWARD -i eth1 -j ACCEPT
$ iptables -A FORWARD -o eth1 -j ACCEPT
```

   - **POSTROUTING**:

```
$ iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
$ iptables -t nat -A POSTROUTING -d $HA_PROXY_IP -s $INTERNAL_NETWORK_RANGE \
    -p tcp --dport 80 -j SNAT --to $GATEWAY_INTERNAL_IP
$ iptables -t nat -A POSTROUTING -d $HA_PROXY_IP -s $INTERNAL_NETWORK_RANGE \
    -p tcp --dport 443 -j SNAT --to $GATEWAY_INTERNAL_IP
```

   - **PREROUTING**:

```
$ iptables -t nat -A PREROUTING -d $GATEWAY_EXTERNAL_IP -p tcp --dport \
    25555 -j DNAT --to $PIVOTALCF_IP
$ iptables -t nat -A PREROUTING -d $GATEWAY_EXTERNAL_IP -p tcp --dport \
    443 -j DNAT --to $HA_PROXY_IP
$ iptables -t nat -A PREROUTING -d $GATEWAY_EXTERNAL_IP -p tcp --dport \
    80 -j DNAT --to $HA_PROXY_IP
$ iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 8443 -j DNAT \
   --to $PIVOTALCF_IP:443
$ iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j DNAT \
   --to $HA_PROXY_IP:80
$ iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 8022 -j DNAT \
   --to $PIVOTALCF_IP:22
$ iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 8080 -j DNAT \
   --to $PIVOTALCF_IP:80
```

6. Run the following command to save the iptables:

```
$ service iptables save
```

For more information about administering IP tables with `iptables`, refer to the iptables documentation.

Verify PCF Resolves DNS Entries Behind a Firewall

When you install PCF in an environment that uses a strong firewall, the firewall might block DNS resolution. For example, if you use xip.io to test your DNS configuration, the tests will fail without warning if the firewall prevents PAS from accessing `*.xip.io`.

To verify that PAS can correctly resolve DNS entries:

1. SSH into the Pivotal Ops Manager VM.
   For more information, refer to the SSH into Ops Manager section of the Advanced Troubleshooting with the BOSH CLI topic.

2. Run any of the following network administration commands with the IP address of the VM:
   - `nslookup`
   - `dig`
   - `host`
   - The appropriate `traceroute` command for your OS

3. Review the output of the command and fix any blocked routes.
   If the output displays an error message, review the firewall logs to determine which blocked route or routes you need to clear.

4. Repeat steps 1-3 with the Ops Manager Director VM and the HAProxy VM.

IaaS Permissions Guidelines

This topic describes practices recommended by Pivotal for creating secure IaaS user roles.

Pivotal Cloud Foundry (PCF) is an automated platform that connects to IaaS providers such as AWS and OpenStack. This connectivity typically requires accounts with appropriate permissions to act on behalf of the operator to access IaaS functionality such as creating virtual machines (VMs), managing networks and storage, and other related services.

Ops Manager and Pivotal Application Service (PAS) can be configured with IaaS users in different ways depending on your IaaS. Other product tiles and services might also use their own IaaS credentials. Refer to the documentation for those product tiles or services to configure them securely.

Least Privileged Users (LPUs)

Pivotal recommends following the principle of least privilege by scoping privileges to the most restrictive permissions possible for a given role. In the event that someone gains access to credentials by mistake or through malicious intent, LPUs limit the scope of the breach. Pivotal recommends following best practices for the particular IaaS you are deploying.

AWS Guidelines

See the recommendations detailed in the AWS Permissions Guidelines topic.

Azure Guidelines

See the permissions recommendations in installation instructions, and use the minimum permissions necessary when creating your service principal.

GCP Guidelines

For GCP, Pivotal recommends using two different accounts with the least privilege.

Use one account with the minimum permissions required to create desired GCP resources in your GCP project, then create a separate service account with the minimum permissions required to deploy PCF components such as Pivotal Ops Manager and PAS.

OpenStack Guidelines

See the installation instructions and follow the least privileged user configuration for tenants and identity.

vSphere Guidelines

See the vCenter permissions recommendations in the Installing Pivotal Cloud Foundry on vSphere topic.

Pivotal

PCF on AWS Requirements

Page last updated:

This topic lists the requirements for installing Pivotal Cloud Foundry (PCF) on Amazon Web Services (AWS).

General Requirements

The following are general requirements for deploying and managing a PCF deployment with Ops Manager and Pivotal Application Service (PAS):

- A wildcard DNS record that points to your router or load balancer. Alternatively, you can use a service such as xip.io. For example, `203.0.113.0.xip.io`.

  PAS gives each application its own hostname in your app domain. With a wildcard DNS record, every hostname in your domain resolves to the IP address of your router or load balancer, and you do not need to configure an A record for each app hostname. For example, if you create a DNS record `*.example.com` pointing to your load balancer or router, every application deployed to the `example.com` domain resolves to the IP address of your router.
- At least one wildcard TLS certificate that matches the DNS record you set up above, `*.example.com`.
- Sufficient IP allocation:
  - One static IP address for either HAProxy or one of your gorouters
  - One IP address for each VM instance
  - An additional IP address for each compilation worker
  So the formula for total IPs needed is `IPs needed = static IPs + VM instances + compilation workers`

  > **Note**: Pivotal recommends that you allocate at least 36 dynamic IP addresses when deploying Ops Manager and PAS. BOSH requires additional dynamic IP addresses during installation to compile and deploy VMs, install PAS, and connect to services.
- One or more NTP servers if not already provided by your IaaS.
- (**Recommended**) A network without DHCP available for deploying the PAS VMs.

  > **Note**: If you have DHCP, refer to the Troubleshooting Guide to avoid issues with your installation.
- (**Optional**) External storage. When you deploy PCF, you can select internal file storage or external file storage, either network-accessible or IaaS-provided, as an option in the PAS tile. Pivotal recommends using external storage whenever possible. See Upgrade Considerations for Selecting File Storage in Pivotal Cloud Foundry for a discussion of how file storage location affects platform performance and stability during upgrades.
- (**Optional**) External databases. When you deploy PCF, you can select internal or external databases for the BOSH Director and for PAS. Pivotal recommends using external databases in production deployments.
- (**Optional**) External user stores. When you deploy PCF, you can select a SAML user store for Ops Manager or a SAML or LDAP user store for PAS, to integrate existing user accounts.
- The most recent version of the Cloud Foundry Command Line Interface (cf CLI).

AWS Requirements

The following are the minimum resource requirements for maintaining a HA Pivotal Cloud Foundry (PCF) deployment with Ops Manager and Pivotal Application Service (PAS) on Amazon Web Services infrastructure:

- 3 Elastic Load Balancer
- 1 Relational Database Service. Pivotal recommends at least a db.m3.xlarge instance with 100 GB of allocated storage.
- 5 S3 Buckets
- EC2 Instances:
  - 24 t2.micro
  - 4 t2.small
  - 4 t2.medium
  - 5 m4.large
  - 3 m3.xlarge
  - 3 m4.xlarge
  - 3 m3.2xlarge

- 4 c4.xlarge

By default, PAS deploys the number of VM instances required to run a highly available (HA) configuration of PCF. If you are deploying a test or sandbox PCF that does not require HA, you can scale down the number of instances in your deployment. For information about the number of instances required to run a minimal, non-HA PCF deployment, see Scaling PAS.

You must have the following to install PCF on AWS:

- An AWS account that can accommodate the minimum resource requirements for a PCF installation.
- The appropriate region selected within your AWS account. For help selecting the correct region for your deployment, see the AWS documentation about regions and availability zones.
- The AWS CLI installed on your machine and configured with user credentials that have admin access to your AWS account.
- Sufficiently high instance limits, or no instance limits, on your AWS account. Installing PCF requires more than the default 20 concurrent instances.
- A key pair to use with your PCF deployment. For more information, see the AWS documentation about creating a key pair.
- A registered wildcard domain for your PCF installation. You need this registered domain when configuring your SSL certificate and Cloud Controller. For more information, see the AWS documentation about Creating a Server Certificate.
- An SSL certificate for your PCF domain. This can be a self-signed certificate, but Pivotal recommends using a self-signed certificate for testing and development. You should obtain a certificate from your Certificate Authority for use in production. For more information, see the AWS documentation about SSL certificates.

Certificate Requirements on AWS

If you are deploying PCF on AWS, then the certificate that you configure in PAS must match the certificate that you upload to AWS as a prerequisite to PCF deployment.

See Certificate Requirements for general certificate requirements for deploying PCF.

AWS Security Documents

- AWS Identity and Access Management guide
  This guide is a reference for AWS' Identity and Access Management (IAM) features. If you're new to AWS, start here.

- AWS identity documentation
- AWS credential documentation
  This documentation provides a general definition of IAM terms and provide best practices to help you manage IaaS users and permissions.

AWS Permissions Guidelines

Pivotal recommends following the principle of least privilege by scoping privileges to the most restrictive permissions possible for a given role. SeeAWS Permissions Guidelines for recommendations on how to create and scope AWS accounts for PCF.

Delete PCF on AWS

You can use the AWS console to remove an installation of all components, but retain the objects in your bucket for a future deployment:

- Deleting an AWS Installation from the Console

AWS Permissions Guidelines

Pivotal recommends that you minimize the use of master account credentials by creating an IAM role and instance profile with the minimum required EC2, VPC, and EBS credentials.

In addition, Pivotal recommends that you follow AWS account security best practices such as disabling root keys, using multi-factor authentication on the root account, and using CloudTrail for auditing API actions.

For more Amazon-specific best practices, refer to the following Amazon documentation:

- IAM Roles Best Practices
- AWS Security Best Practices Whitepaper
- AWS Well-Architected Framework

Configuring Ops Manager on AWS Deployed Manually

Page last updated:

This topic describes how to manually configure the Amazon Web Services (AWS) components that you need to run Pivotal Cloud Foundry (PCF) on AWS.

To deploy PCF on AWS, you must perform the procedures in this topic to create objects in the AWS Management Console that PCF requires.

To view the list of AWS objects created by the procedures in this topic, see the Required AWS Objects section.

After completing the procedures in this topic, proceed to Manually Configuring Ops Manager Director for AWS to continue deploying PCF.

Step 1: File a Ticket

Log in to the AWS Management Console, and file a ticket with Amazon to ensure that your account can launch more than the default 20 instances. In the ticket, ask for a limit of 50 `t2.micro` instances and 20 `c4.large` instances in the region you are using.

> **Note**: To deploy PCF to AWS GovCloud (US), log in to the AWS GovCloud (US) Console instead of the standard AWS Management Console and select the `us-gov-west-1` region.

> **Note**: To deploy PCF to AWS China, set up an AWS China account and contact Pivotal Support.

You can check the limits on your account by visiting the EC2 Dashboard on the AWS Management Console and clicking **Limits** on the left navigation.

Step 2: Create S3 Buckets

1. Navigate to the S3 Dashboard.

2. Perform the following steps to create five S3 buckets:
   - Click **Create Bucket**
   - For **Bucket name**, enter `pcf-ops-manager-bucket`.
   - For **Region**, select your region.
   - Click **Next** three times.
   - Click **Create bucket**.
   - Repeat the above steps to create four more S3 buckets: `pcf-buildpacks-bucket`, `pcf-packages-bucket`, `pcf-resources-bucket`, and `pcf-droplets-bucket`.

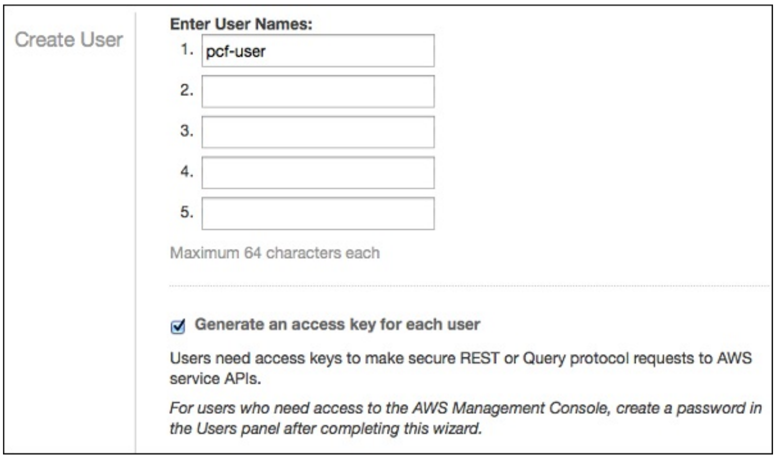
Step 3: Create an IAM User for PCF

Perform the following steps to create an Amazon Identity and Access Management (IAM) user with the minimal permissions necessary to run and install PCF:

1. Select **Identity & Access Management** to access the IAM Dashboard.

2. Select **Users>Create New Users**.

3. Enter a user name, such as `pcf-user`.



4. Ensure that the **Generate an access key for each user** checkbox is selected.

> **Note**: If you prefer to create your keys locally and import them into AWS, see the Amazon documentation.

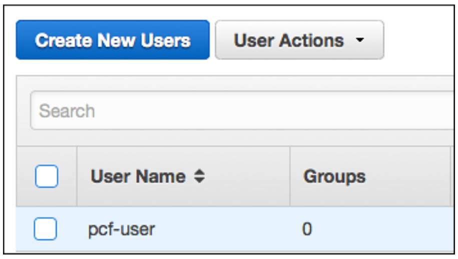
5. Click **Create**.

6. Click **Download Credentials** to download the user security credentials.

> **WARNING**: The `credentials.csv` contains the IDs for your user security access key and secret access key. Keep the `credentials.csv` file for your currently active key pairs in a secure directory. You cannot recover a lost key pair.

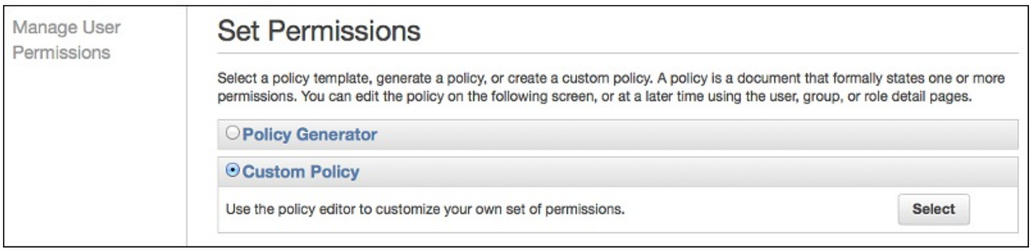
7. Click **Close**.

8. On the **User** dashboard, click the user name to access the user details page.



9. In the **Inline Policies** region, click the down arrow to display the inline policies. Click the **click here** link to create a new policy.

10. On the **Set Permissions** page, click **Custom Policy** and click **Select**.



11. On the **Review Policy** page, enter `pcf-iam-policy` in **Policy Name**.

12. Copy and paste the policy document included in the [Pivotal Cloud Foundry for AWS Policy Document]() topic into the **Policy Document** field.

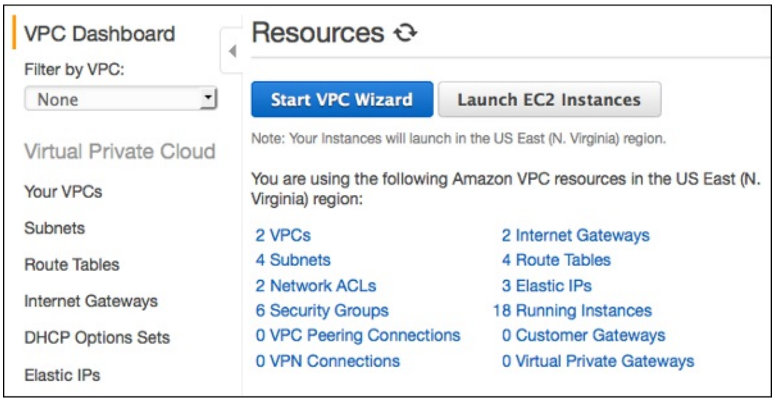
13. Ensure that the **Use autoformatting for policy editing** checkbox is selected.

14. Click **Apply Policy** and review. The **Inline Policies** region now displays a list of available policies and actions.

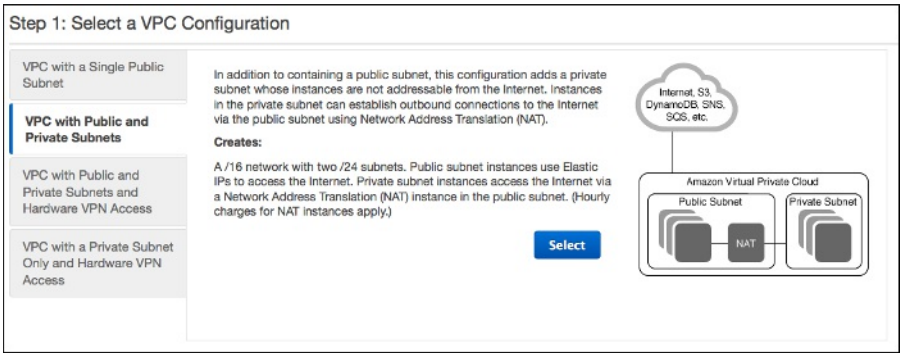
Step 4: Create a VPC

1. Navigate to the VPC Dashboard.

2. Click **Start VPC Wizard**.



3. Select **VPC with Public and Private Subnets** and click **Select**.



4. Specify the following details for your VPC:
   - **IPv4 CIDR block**: Enter `10.0.0.0/16`.
   - **IPv6 CIDR block**: Select **No IPv6 CIDR Block**.
   - **VPC name:** `pcf-vpc`.
   - **Public subnet's IPv4 CIDR**: Enter `10.0.0.0/24`.
   - Set the **Availability Zone** fields for both subnets to `REGION-#a`. For example, **us-west-2a**.
   - **Public subnet name**: Enter `pcf-public-subnet-az0`.
   - **Private subnet's IPv4 CIDR**: Enter `10.0.16.0/28`.
   - **Private subnet name**: Enter `pcf-management-subnet-az0`.
   - Click **Use a NAT instance instead** and under **Specify the details of your NAT instance**, set the **Instance type** to **t2.medium** and select the `pcf-ops-manager-key` SSH key you created for **Key Pair name**.
   - **Enable DNS hostnames**: Click **Yes**.
   - **Hardware tenancy**: Select **Default**.
   - **Enable ClassicLink**: Select **No**.
   - Click **Create VPC**.

5. After the VPC is successfully created, click **Subnets** in the left navigation.

6. Click **Create Subnet**.

7. Add the following subnets to the `pcf-vpc` VPC:

Note: You created the first two subnets in the previous step: `pcf-public-subnet-az0` and `pcf-management-subnet-az0`.

Name	AZ	IPv4 CIDR block
`pcf-public-subnet-az1`	`REGION-#b` (for example, `us-west-2b`)	`10.0.1.0/24`
`pcf-public-subnet-az2`	`REGION-#c` (for example, `us-west-2c`)	`10.0.2.0/24`
`pcf-management-subnet-az1`	`REGION-#b` (for example, `us-west-2b`)	`10.0.16.16/28`
`pcf-management-subnet-az2`	`REGION-#c` (for example, `us-west-2c`)	`10.0.16.32/28`
`pcf-ert-subnet-az0`	`REGION-#a` (for example, `us-west-2a`)	`10.0.4.0/24`
`pcf-ert-subnet-az1`	`REGION-#b` (for example, `us-west-2b`)	`10.0.5.0/24`
`pcf-ert-subnet-az2`	`REGION-#c` (for example, `us-west-2c`)	`10.0.6.0/24`
`pcf-services-subnet-az0`	`REGION-#a` (for example, `us-west-2a`)	`10.0.8.0/24`
`pcf-services-subnet-az1`	`REGION-#b` (for example, `us-west-2b`)	`10.0.9.0/24`
`pcf-services-subnet-az2`	`REGION-#c` (for example, `us-west-2c`)	`10.0.10.0/24`
`pcf-rds-subnet-az0`	`REGION-#a` (for example, `us-west-2a`)	`10.0.12.0/24`
`pcf-rds-subnet-az1`	`REGION-#b` (for example, `us-west-2b`)	`10.0.13.0/24`
`pcf-rds-subnet-az2`	`REGION-#c` (for example, `us-west-2c`)	`10.0.14.0/24`

Step 5: Configure a Security Group for Ops Manager

1. Return to the EC2 Dashboard.
2. Select **Security Groups>Create Security Group**.
3. For **Security group name**, enter `pcf-ops-manager-security-group`.
4. For **Description**, enter a description to identify this security group.
5. For **VPC**, select the VPC where you want to deploy Ops Manager.
6. Click the **Inbound** tab and add rules according to the table below.

   Note: Pivotal recommends limiting access to Ops Manager to IP ranges within your organization, but you may relax the IP restrictions after configuring authentication for Ops Manager.

Type	Protocol	Port Range	Source
HTTP	TCP	80	My IP
HTTPS	TCP	443	My IP
SSH	TCP	22	My IP
BOSH Agent	TCP	6868	10.0.0.0/16
BOSH Director	TCP	25555	10.0.0.0/16

7. Click **Create**.

Step 6: Configure a Security Group for PCF VMs

1. From the **Security Groups** page, click **Create Security Group** to create another security group.
2. For **Security group name**, enter `pcf-vms-security-group`.
3. For **Description**, enter a description to identify this security group.
4. For **VPC**, select the VPC where you want to deploy the PCF VMs.
5. Click the **Inbound** tab and add rules for all traffic from your public and private subnets to your private subnet, as the table and image show. This rule

configuration does the following:
  - Enables BOSH to deploy PAS and other services.
  - Enables application VMs to communicate through the router.
  - Allows the load balancer to send traffic to Pivotal Application Service (PAS).

Type	Protocol	Port Range	Source	
All traffic	All	0 - 65535	Custom IP	10.0.0.0/16

6. Click **Create**.

Step 7: Configure a Security Group for the Web ELB

1. From the **Security Groups** page, click **Create Security Group** to create another security group.

2. For **Security group name**, enter `pcf-web-elb-security-group`.

3. For **Description**, enter a description to identify this security group.

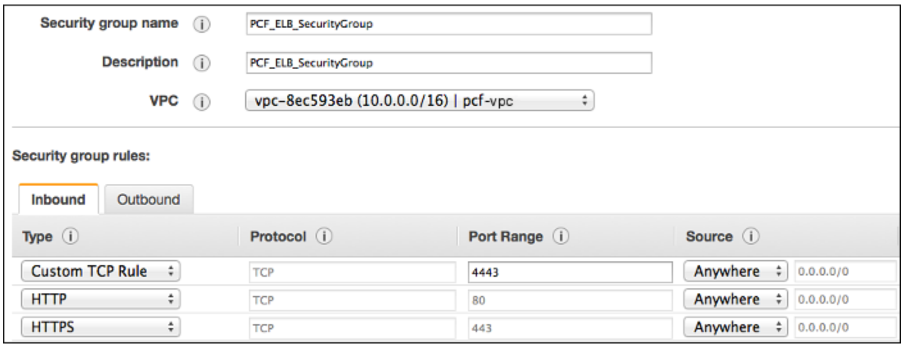
4. For **VPC**, select the VPC where you want to deploy this Elastic Load Balancer (ELB).

5. Click the **Inbound** tab and add rules to allow traffic to ports `80`, `443`, and `4443` from `0.0.0.0/0`, as the table and image show.

> **Note**: You can change the `0.0.0.0/0` to be more restrictive if you want finer control over what can reach the PAS. This security group governs external access to the PAS from applications such as the cf CLI and application URLs.

Type	Protocol	Port Range	Source	
Custom TCP rule	TCP	4443	Anywhere	0.0.0.0/0
HTTP	TCP	80	Anywhere	0.0.0.0/0
HTTPS	TCP	443	Anywhere	0.0.0.0/0

6. Click **Create**.



Step 8: Configure a Security Group for the SSH ELB

1. From the **Security Groups** page, click **Create Security Group** to create another security group.

2. For **Security group name**, enter `pcf-ssh-elb-security-group`.

3. For **Description**, enter a description to identify this security group.

4. For **VPC**, select the VPC where you want to deploy this ELB.

5. Click the **Inbound** tab and add the following rule:

Type	Protocol	Port Range	Source	
Custom TCP rule	TCP	2222	Anywhere	0.0.0.0/0

6. Click **Create**.

Step 9: Configure a Security Group for the TCP ELB

1. From the **Security Groups** page, click **Create Security Group** to create another security group.

2. For **Security group name**, enter `pcf-tcp-elb-security-group`.

3. For **Description**, enter a description to identify this security group.

4. For **VPC**, select the VPC where you want to deploy this ELB.

5. Click the **Inbound** tab and add the following rule:

Type	Protocol	Port Range	Source	
Custom TCP rule	TCP	1024 - 1123	Anywhere	0.0.0.0/0

6. Click **Create**.

Step 10: Configure a Security Group for the Outbound NAT

1. From the **Security Groups** page, click **Create Security Group** to create another security group.

2. For **Security group name**, enter `pcf-nat-security-group`.

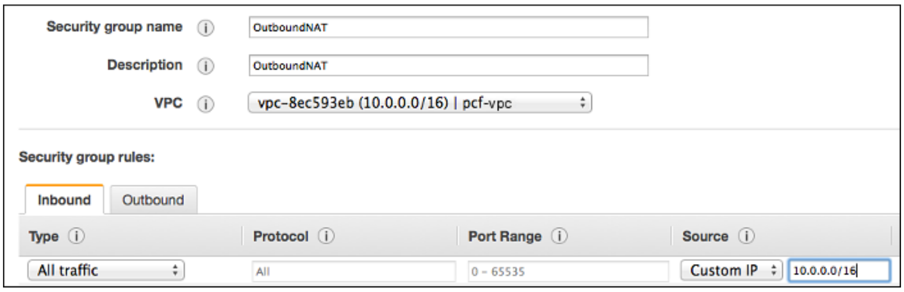
3. For **Description**, enter a description to identify this security group.

4. For **VPC**, select the VPC where you want to deploy the Outbound NAT.

5. Click the **Inbound** tab and add a rule to allow all traffic from your VPCs, as the table and image show.

Type	Protocol	Port Range	Source	
All traffic	All	All	Custom IP	10.0.0.0/16

6. Click **Create**.



Step 11: Configure a Security Group for MySQL

Note: If you plan to use an internal database, skip this step. If you are using RDS, you must configure a security group that enables the Ops Manager VM and Ops Manager Director VM to access the database.

1. From the **Security Groups** page, click **Create Security Group** to create another security group.

2. For **Security group name**, enter `MySQL`.

3. For **Description**, enter a description to identify this security group.

4. For **VPC**, select the VPC where you want to deploy MySQL.

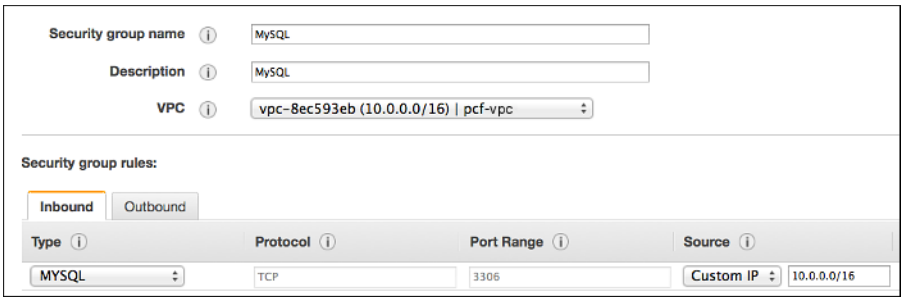
5. Click the **Inbound** tab. Add a rule of type `MySQL` and specify the subnet of your VPC in **Source**, as the table and image show.

Type	Protocol	Port Range	Source	
MySQL	TCP	3306	Custom IP	10.0.0.0/16

6. Click the **Outbound** tab. Add a rule of type `All traffic` and specify the subnet of your VPC in **Destination**, as the table and image show.

Type	Protocol	Port Range	Destination	
All traffic	All	All	Custom IP	10.0.0.0/16

7. Click **Create**.



Step 12: Launch a Pivotal Ops Manager AMI

1. Navigate to the **Pivotal Cloud Foundry Operations Manager** section of Pivotal Network.

2. Select the version of PCF you want to install and click **Pivotal Cloud Foundry Ops Manager for AWS** to download the PDF.

3. Open the PDF and identify the AMI ID for your region.

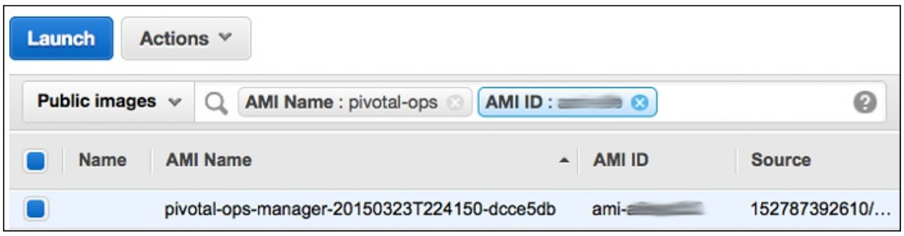
4. Return to the EC2 Dashboard.

5. Click **AMIs**.

6. Click **Owned by me** and select **Public image**.

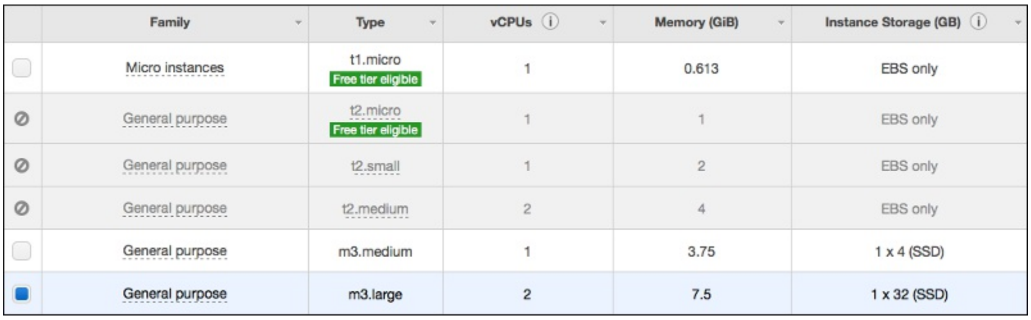
7. Paste the AMI ID for your region into the search bar and press enter.

> **Note**: There is a different AMI for each region. If you cannot locate the AMI for your region, verify that you have set your AWS Management Console to your desired region. If you still cannot locate the AMI, log in to the Pivotal Network and file a support ticket.

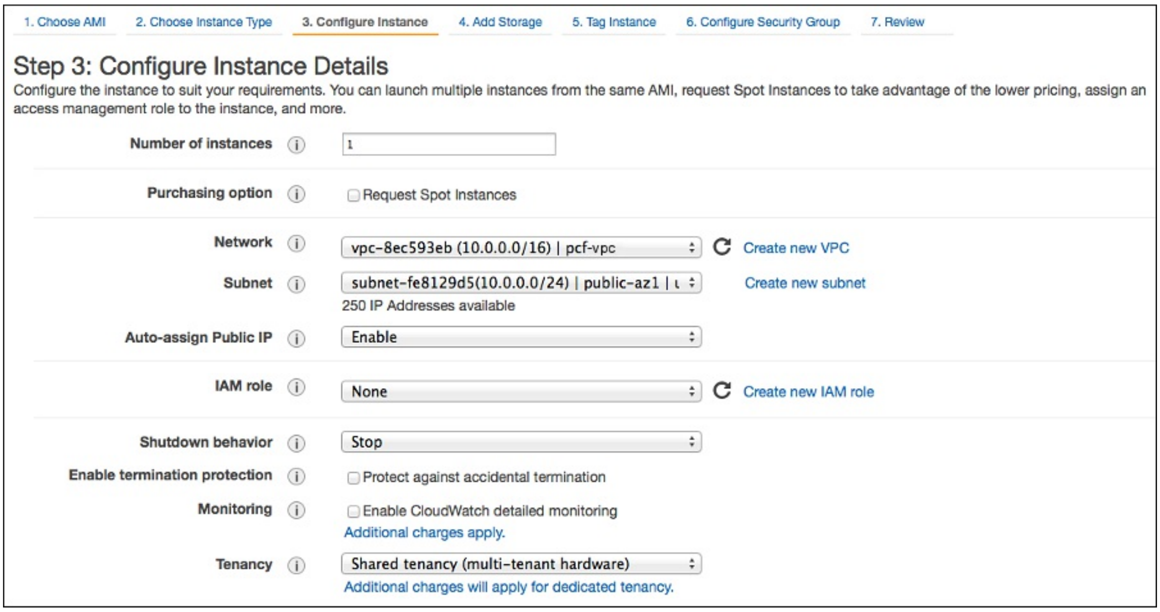


8. Select the Ops Manager API and click **Launch**.

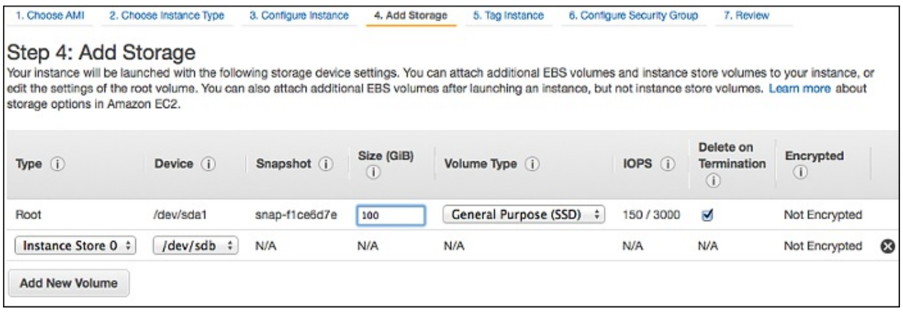
9. Choose **m3.large** for your instance type and click **Next: Configure Instance Details**.



10. Configure the following for your instance:
    - **Network**: Select the VPC that you created.
    - **Subnet**: Select `pcf-public-subnet-az0`.
    - **Auto-assign for Public IP**: Select **Enable**.
    - **IAM role**: Select the IAM role associated with your pcf-user profile. If you have not created one, click **Create new IAM role** and follow the Guidelines for Creating User Roles on AWS.
    - For all other fields, accept the default values.



11. Click **Next: Add Storage** and adjust the **Size (GiB)** value. The default persistent disk value is 50 GB. Pivotal recommends increasing this value to a minimum of 100 GB.



12. Click **Next: Tag Instance**

13. On the **Add Tags** page, add a tag with the key `Name` and value `pcf-ops-manager`.

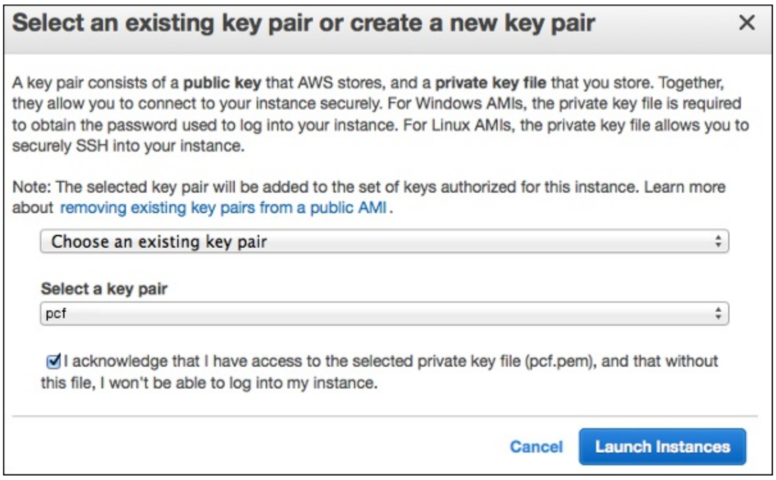
14. Click **Next: Configure Security Group**.

15. Select the `pcf-ops-manager-security-group` that you created in Step 5: Configure a Security Group for Ops Manager.

16. Click **Review and Launch** and confirm the instance launch details.

17. Click **Launch**.

18. Select the `pcf-ops-manager-key` key pair, confirm that you have access to the private key file, and click **Launch Instances**. You use this key pair to access the Ops Manager VM.



19. Click **View Instances** to access the **Instances** page on the EC2 Dashboard.

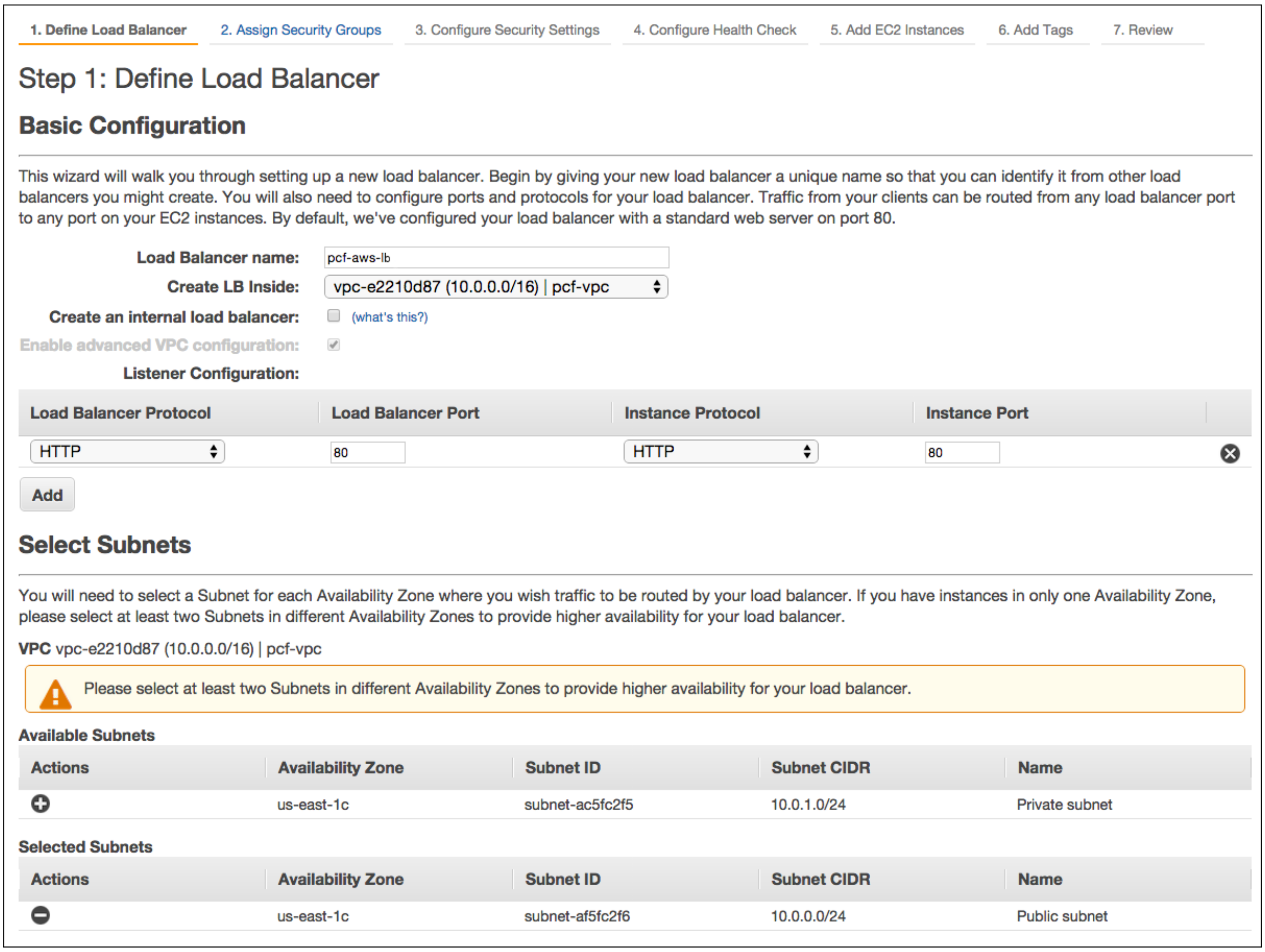
Step 13: Create Web Load Balancer

1. On the EC2 Dashboard, click **Load Balancers**.

2. Click **Create Load Balancer**.

3. Select **Classic Load Balancer**.

4. Configure the load balancer with the following information:
   - **Load Balancer name**: Enter `pcf-web-elb`.
   - **Create LB Inside**: Select the `pcf-vpc` VPC that you created in Step 4: Create a VPC.
   - Ensure that the **Create an internal load balancer** checkbox is not selected.



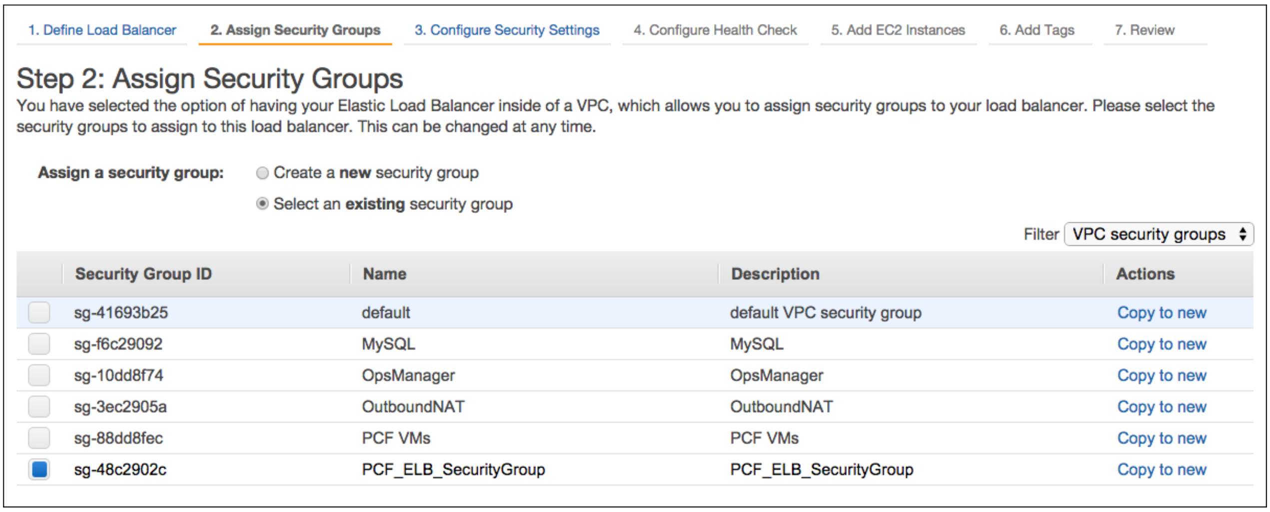
5. Under **Listener Configuration**, add the following rules:

Load Balancer Protocol	Load Balancer Port	Instance Protocol	Instance Port
HTTP	80	HTTP	80
HTTPS	443	HTTP	80
SSL	4443	TCP	80

If you require traffic between the load balancer and HAProxy or the Gorouter to be encrypted with TLS, consider that the AWS Classic load balancers do not support recommended TLS cipher suites. See Securing Traffic into Cloud Foundry for details.

6. Under **Select Subnets**, select the public subnets you configured in Step 4: Create a VPC, and click **Next: Assign Security Groups**.

7. On the **Assign Security Groups** page, select the security group `pcf-elb-security-group` you configured in Step 7: Configure a Security Group for the Web ELB, and click **Next: Configure Security Settings**.

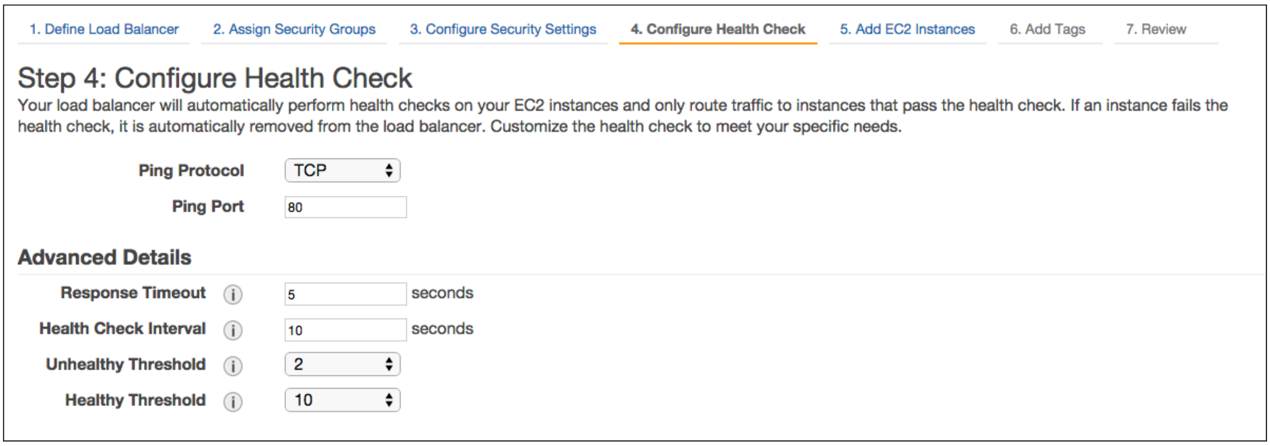


8. On the **Configure Security Settings** page, select **Upload a new SSL certificate to AWS Identity and Access Management (IAM)**. For a production environment, use a certificate from a Certificate Authority. For a development environment, use a self-signed certificate.

> **Note**: In this configuration, SSL traffic will be decrypted at the load balancer and then sent to the Gorouter instances in the VPC.

9. On the **Configure Health Check** page, enter the following values:
   - **Ping Protocol**: Select **HTTP**.
   - **Ping Port**: Set to `8080`.
   - **Ping Path**: Set to `\health`.
   - **Interval**: Set to `5` seconds.
   - **Response Timeout**: Set to `3` seconds.
   - **Unhealthy threshold**: Set to `3`.
   - **Health threshold**: Set to `6`.

10. Click **Next: Add EC2 Instances**.



11. Accept the defaults on the **Add EC2 Instances** page and click **Next: Add Tags**.

12. Accept the defaults on the **Add Tags** page and click **Review and Create**.

13. Review and confirm the load balancer details, and click **Create**.

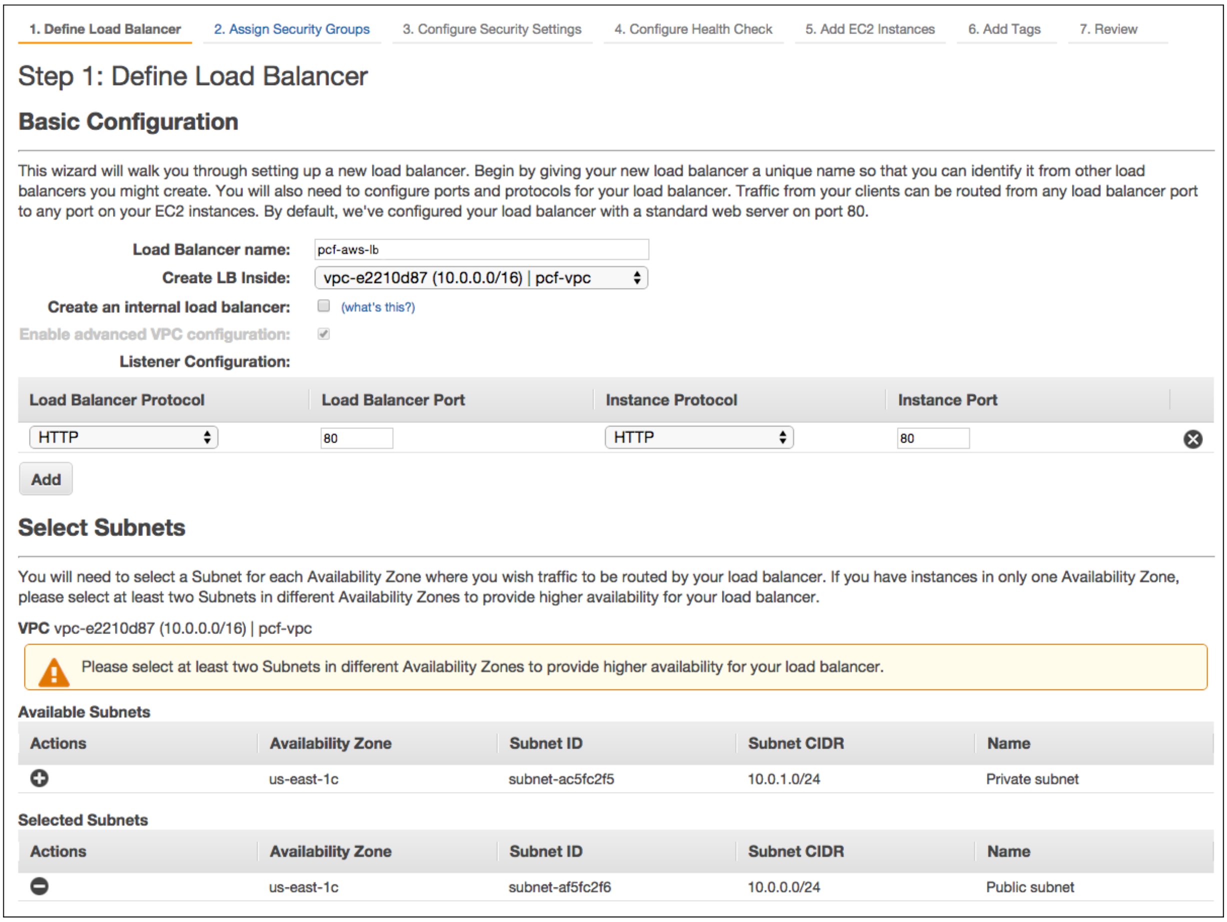
Step 14: Create SSH Load Balancer

1. From the **Load Balancers** page, click **Create Load Balancer**.

2. Select **Classic Load Balancer**.

3. Configure the load balancer with the following information:

- **Load Balancer name**: Enter `pcf-ssh-elb` .
- **Create LB Inside**: Select the `pcf-vpc` VPC that you created in Step 4: Create a VPC.
- Ensure that the **Create an internal load balancer** checkbox is not selected.

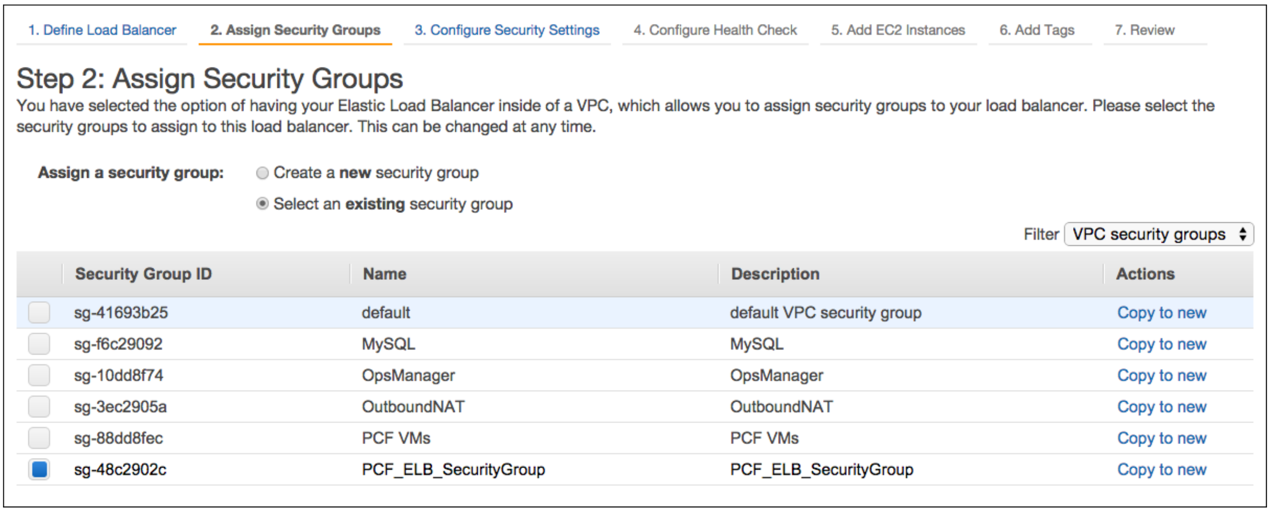


4. Under **Listener Configuration**, add the following rules:

Load Balancer Protocol	Load Balancer Port	Instance Protocol	Instance Port
TCP	2222	TCP	2222

5. Under **Select Subnets**, select the public subnets you configured in Step 4: Create a VPC, and click **Next: Assign Security Groups**.

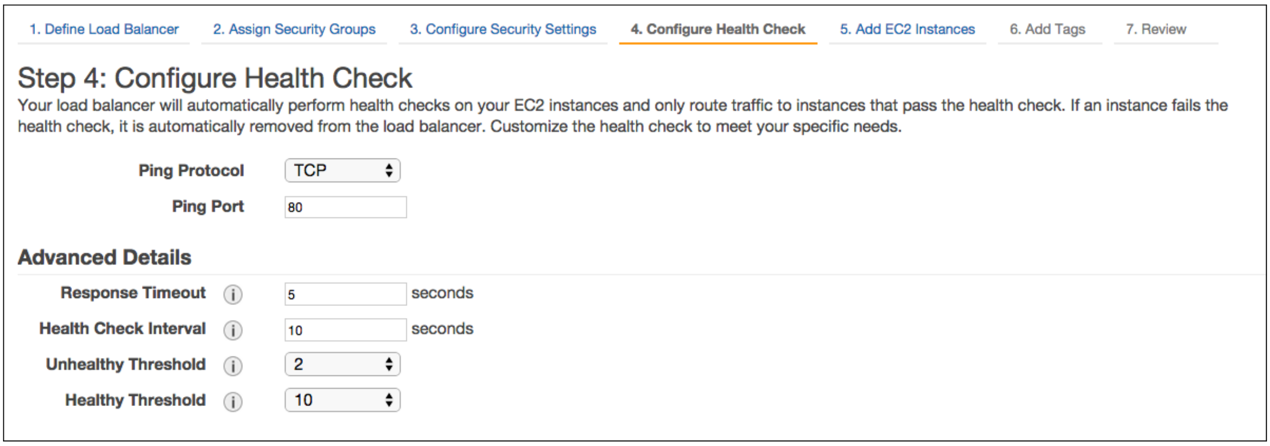
6. On the **Assign Security Groups** page, select the security group `pcf-ssh-elb-security-group` you configured in Step 8: Configure a Security Group for the SSH ELB, and click **Next: Configure Security Settings**.



7. On the **Configure Security Settings** page, ignore the **Improve your load balancer's security** error message and click **Next: Configure Health Check**.

8. On the **Configure Health Check** page, enter the following values:
    - **Ping Protocol**: Select `TCP`.
    - **Ping Port**: Set to `2222`.
    - **Interval**: Set to `5` seconds.
    - **Response Timeout**: Set to `3` seconds.
    - **Unhealthy threshold**: Set to `3`.
    - **Health threshold**: Set to `6`.

9. Click **Next: Add EC2 Instances.**



10. Accept the defaults on the **Add EC2 Instances** page and click **Next: Add Tags**.

11. Accept the defaults on the **Add Tags** page and click **Review and Create**.

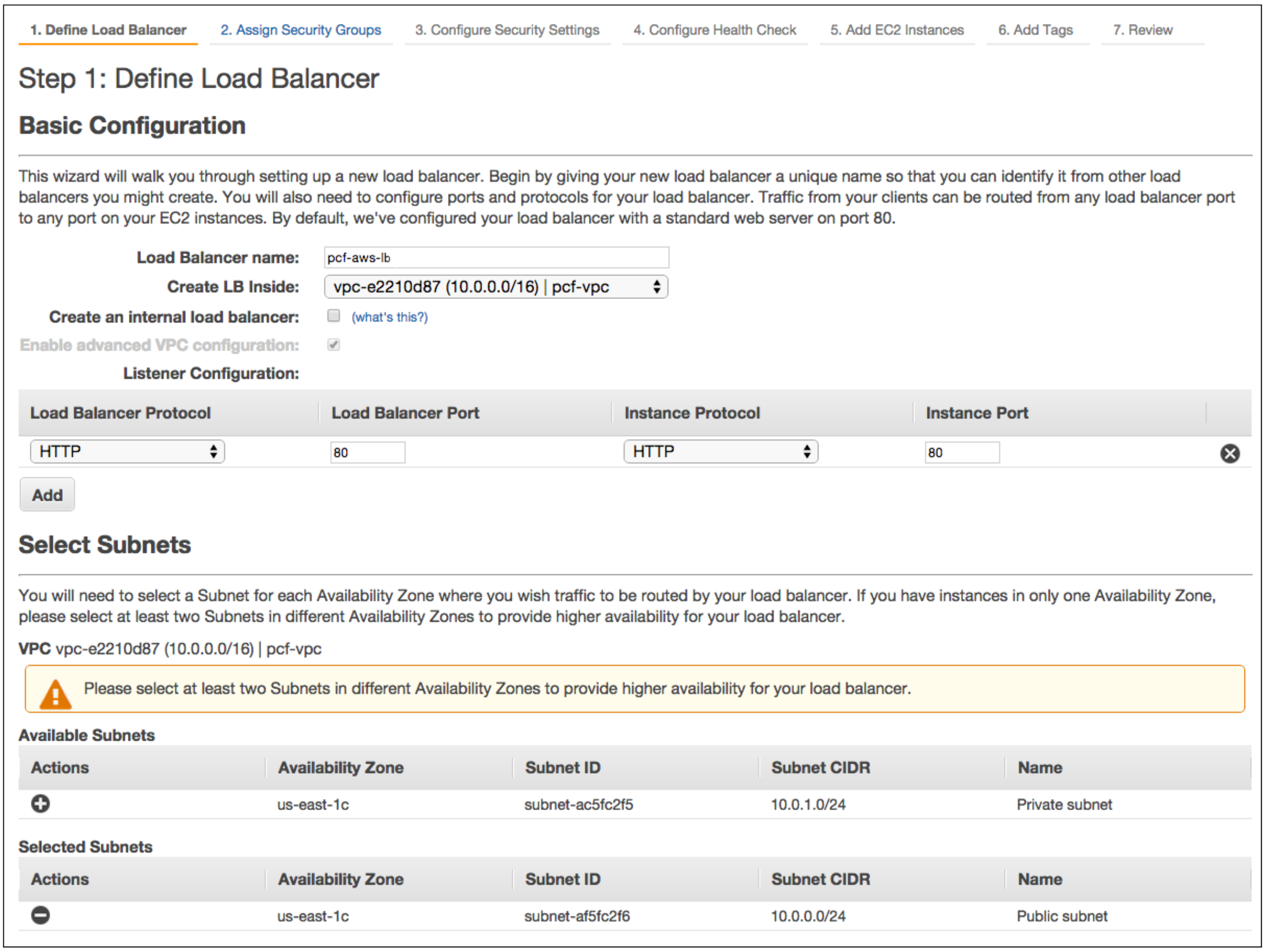
12. Review and confirm the load balancer details, and click **Create**.

Step 15: Create TCP Load Balancer

1. From the **Load Balancers** page, click **Create Load Balancer**.

2. Select **Classic Load Balancer**.

3. Configure the load balancer with the following information:
    - **Load Balancer name**: Enter `pcf-tcp-elb`.
    - **Create LB Inside**: Select the `pcf-vpc` VPC that you created in Step 4: Create a VPC.
    - Ensure that the **Create an internal load balancer** checkbox is not selected.



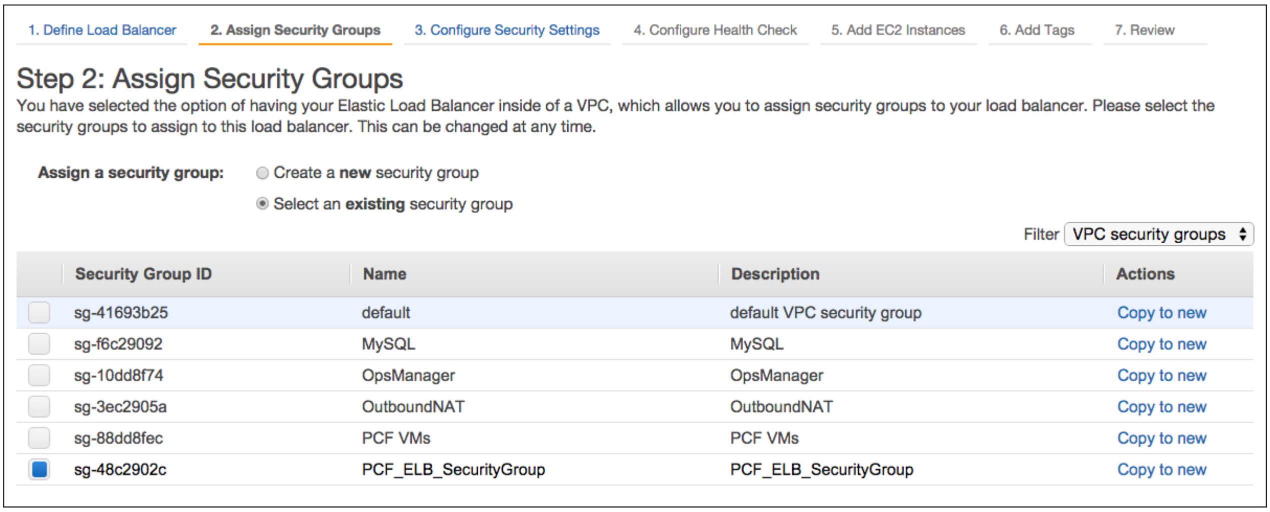
4. Under **Listener Configuration**, add the following rules:

Load Balancer Protocol	Load Balancer Port	Instance Protocol	Instance Port
TCP	1024	TCP	1024
TCP	1025	TCP	1025
TCP	1026	TCP	1026
…	…	…	…
TCP	1123	TCP	1123

The `...` entry above indicates that you must add listening rules for each port between 1026 and 1123.

5. Under **Select Subnets**, select the public subnets you configured in Step 4: Create a VPC, and click **Next: Assign Security Groups**.

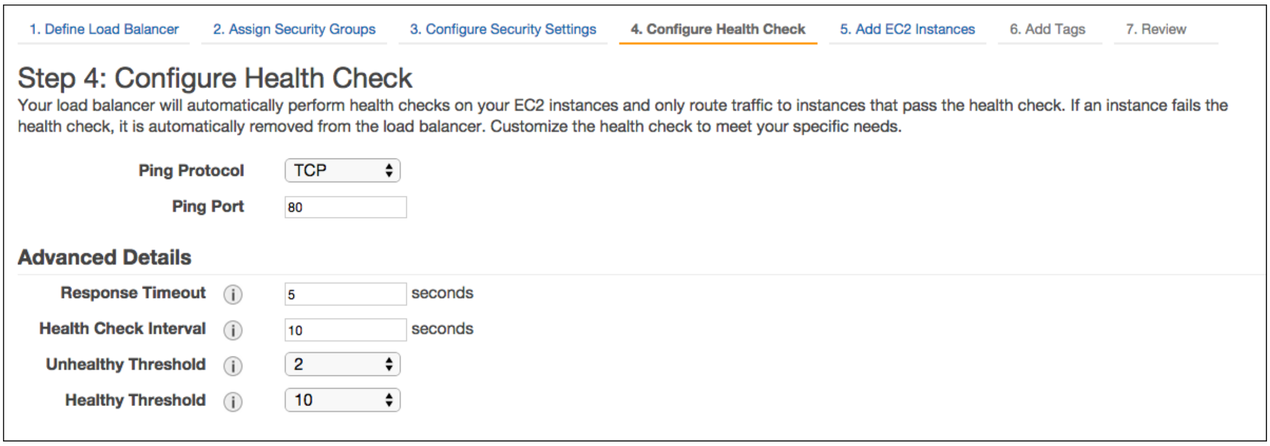
6. On the **Assign Security Groups** page, select the security group `pcf-tcp-elb-security-group` you configured in Step 9: Configure a Security Group for the TCP ELB, and click **Next: Configure Security Settings**.



7. On the **Configure Security Settings** page, ignore the **Improve you load balancer's security** error message and click **Next: Configure Health Check**.

8. On the **Configure Health Check** page, enter the following values:
   - **Ping Protocol**: Select `TCP`.
   - **Ping Port**: Set to `80`.
   - **Interval**: Set to `5` seconds.
   - **Response Timeout**: Set to `3` seconds.
   - **Unhealthy threshold**: Set to `3`.
   - **Health threshold**: Set to `6`.

9. Click **Next: Add EC2 Instances**.



10. Accept the defaults on the **Add EC2 Instances** page and click **Next: Add Tags**.

11. Accept the defaults on the **Add Tags** page and click **Review and Create**.

12. Review and confirm the load balancer details, and click **Create**.

Step 16: Configure DNS Records

1. Perform the following steps for all three of the load balancers you created in previous steps, named `pcf-web-elb`, `pcf-ssh-elb`, and `pcf-tcp-elb`:

   a. From the **Load Balancers** page, select the load balancer.
   b. On the **Description** tab, locate the **Basic Configuration** section and record the **DNS name** of the load balancer.

2. Click **Instances** on the left navigation to view your EC2 instances.

3. Select the `PcfOpsManInstance` instance created by Cloudformation.

4. On the **Description** tab, record the value for **IPv4 Public IP**.

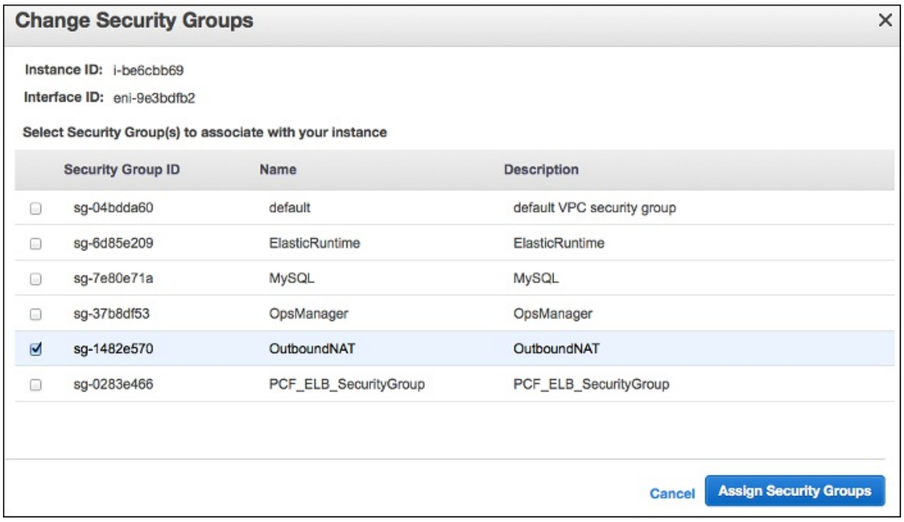
5. Navigate to your DNS provider and create the following CNAME and A records:
    - CNAME: `*.apps.YOUR-SYSTEM-DOMAIN.com` and `*.system.YOUR-SYSTEM-DOMAIN.com` points to the DNS name of the `pcf-web-elb` load balancer.
    - CNAME: `ssh.YOUR-SYSTEM-DOMAIN.com` points to the DNS name of the `pcf-ssh-elb` load balancer.
    - CNAME: `tcp.YOUR-SYSTEM-DOMAIN.com` points to the DNS name of the `pcf-tcp-elb` load balancer.
    - A: `pcf.YOUR-SYSTEM-DOMAIN.com` points to the public IP address of the `pcf-ops-manager` EC2 instance.

Step 17: Secure the NAT Instance

1. On the EC2 Dashboard, click **Instances**.

2. Select the NAT instance, which has an instance type of **t2.medium**.

3. From the **Actions** menu, select **Networking>Change Security Groups**.

4. Change the NAT security group from the default group to the `pcf-nat-security-group` NAT security group that you created in Step 10: Configure 

a Security Group for the Outbound NAT.

5. Click **Assign Security Groups**.

Step 18: Create RDS Subnet Group

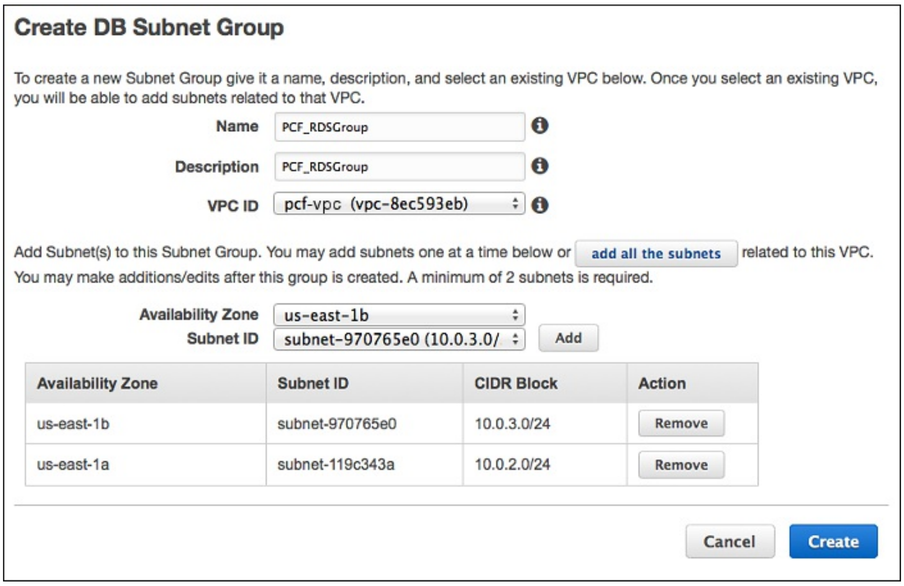
1. Navigate to the RDS Dashboard.

2. Perform the following steps to create a RDS Subnet Group for the two RDS subnets:

    a. Click **Subnet Groups>Create DB Subnet Group.**
    b. Enter the following values:
        - **Name**: Enter `pcf-rds-subnet-group`.
        - **Description**: Enter a description to identify this subnet group.
        - **VPC ID**: Select `pcf-vpc`.
        - **Availability Zone** and **Subnet ID**: Choose the AZ and subnet for `pcf-rds-subnet-az0` and click **Add**.

    c. Repeat the steps above to add `pcf-rds-subnet-az1` and `pcf-rds-subnet-az2` to the group.
    d. Click **Create**.

    The following screenshot shows a completed subnet group.



Note: On the Subnet Group page, you may need to refresh the page to view the new group.

Step 19: Create a MySQL Database Using AWS RDS

Note: You must have an empty MySQL database when you install or reinstall PCF on AWS.

1. Navigate to the RDS Dashboard.

2. Click **Instances>Launch DB Instance** to launch the wizard.

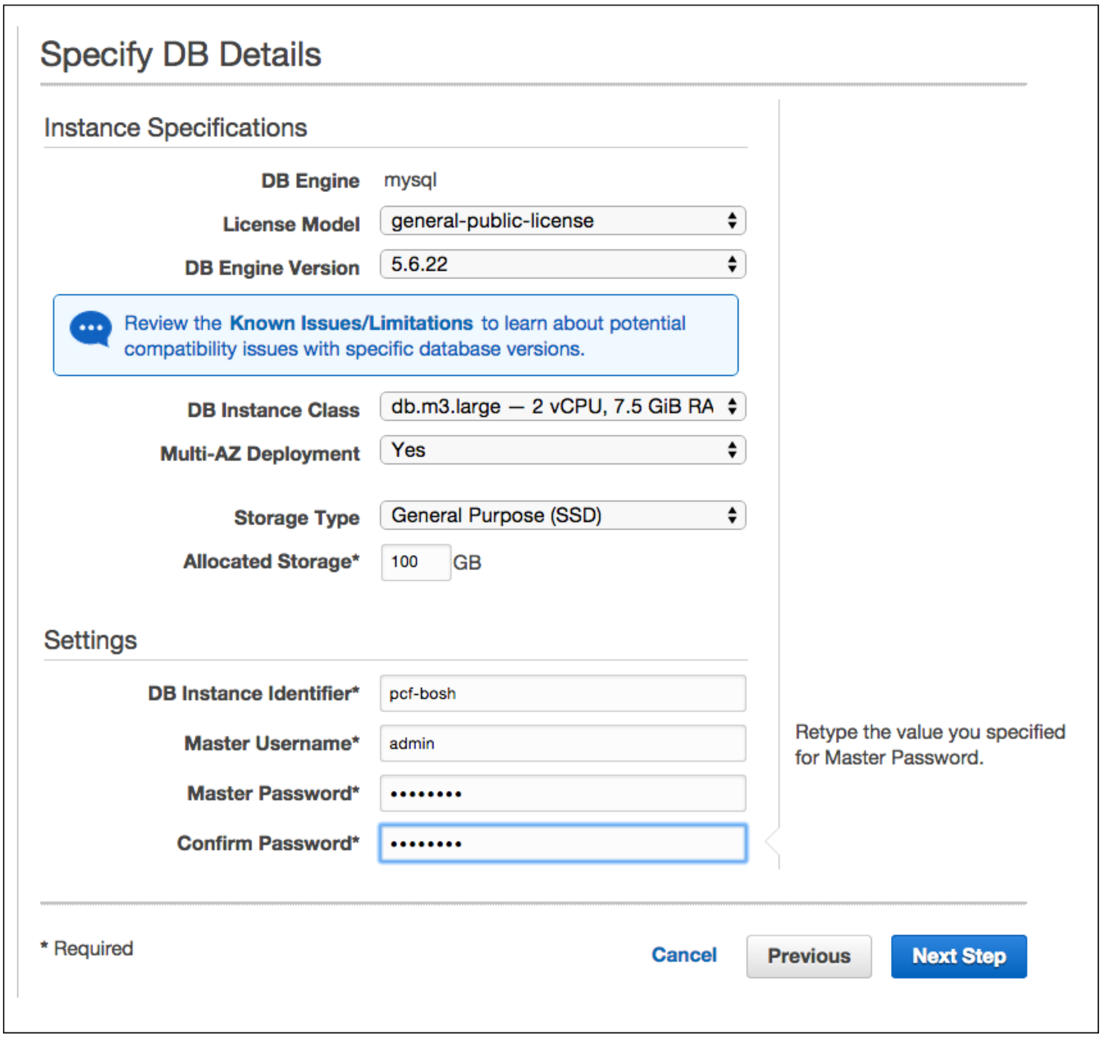
3. Select **MySQL**.

4. Select the **MySQL** radio button under **Production** to create a database for production environments.

5. Click **Next Step**.

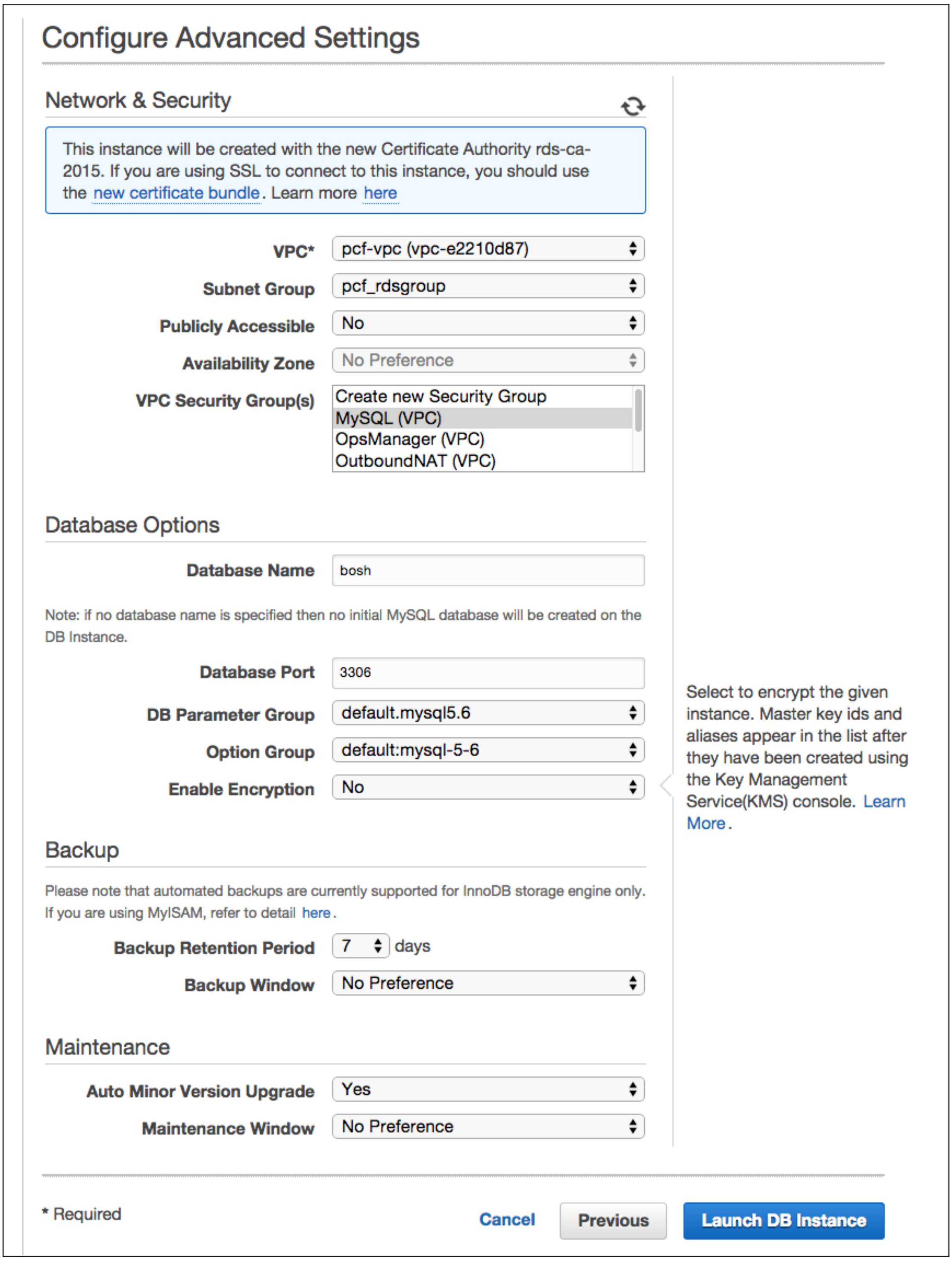
6. Specify the following database details:
   - **DB Instance Class**: Select **db.m3.large - 2 vCPU, 7.5 GiB RAM**.
   - **Multi-AZ Deployment**: Select **Yes**.
   - **Storage Type**: Select `Provisioned IOPS (SSD)`.
   - **Allocated Storage**: Enter **100 GB**.
   - **DB Instance Identifier**: Enter `pcf-ops-manager-director`.
   - Enter a secure **Master Username** and **Master Password**.

   Note: Record the username and password. You need these credentials later when configuring the **Director Config** page in the Ops Manager Director tile.



7. Click **Next Step**.

8. On the **Configure Advanced Settings** page, enter the following values:
   - **VPC**: Select `pcf-vpc`.
   - **Subnet Group**: Select the `pcf-rds-subnet-group` you created in Step 18: Create RDS Subnet Group.
   - **Publicly Accessible**: Select **No**.
   - **VPC Security Groups**: Select the `pcf-rds-security-group` that you created in Step 11: Configure a Security Group for MySQL.
   - **Database Name**: Enter `bosh`.
   - Accept the default values for the remaining fields.



9. Click **Launch DB Instance**. Launching the instance may take several minutes.

10. When the instance has launched, proceed to Manually Configuring Ops Manager Director for AWS to continue deploying PCF.

Required AWS Objects

This section describes the AWS objects you create in the procedures above in order to deploy PCF.

Use this section to determine the resource requirements of PCF on AWS, or to verify that you created the correct resources after completing the procedures above.

S3 Buckets for Ops Manager and PAS

You must create the following S3 buckets from the S3 Dashboard:

- `pcf-ops-manager-bucket`
- `pcf-buildpacks-bucket`
- `pcf-packages-bucket`
- `pcf-resources-bucket`
- `pcf-droplets-bucket`

These buckets must be empty when you install or reinstall PCF.

See Step 2: Create S3 Buckets.

IAM User for PCF

You must create an IAM user for PCF named `pcf-user` from the Identity and Access Management Dashboard, using the policy document included in the Pivotal Cloud Foundry for AWS Policy Document topic.

See Step 3: Create an IAM User for PCF.

Key Pair

You must generate a key pair named `pcf-ops-manager-key` when creating an IAM user.

See Step 3: Create an IAM User for PCF.

VPC (Public and Private Subnets)

You must create a VPC with public and private subnets from the VPC Dashboard.

The following table lists the subnets in CIDR block `10.0.0.0/16`.

Name	AZ	IPv4 CIDR block
`pcf-public-subnet-az0`	`REGION-#a` (for example, `us-west-2a`)	`10.0.0.0/24`
`pcf-public-subnet-az1`	`REGION-#b` (for example, `us-west-2b`)	`10.0.1.0/24`
`pcf-public-subnet-az2`	`REGION-#c` (for example, `us-west-2c`)	`10.0.2.0/24`
`pcf-management-subnet-az0`	`REGION-#a` (for example, `us-west-2a`)	`10.0.16.0/28`
`pcf-management-subnet-az1`	`REGION-#b` (for example, `us-west-2b`)	`10.0.16.16/28`
`pcf-management-subnet-az2`	`REGION-#c` (for example, `us-west-2c`)	`10.0.16.32/28`
`pcf-ert-subnet-az0`	`REGION-#a` (for example, `us-west-2a`)	`10.0.4.0/24`
`pcf-ert-subnet-az1`	`REGION-#b` (for example, `us-west-2b`)	`10.0.5.0/24`
`pcf-ert-subnet-az2`	`REGION-#c` (for example, `us-west-2c`)	`10.0.6.0/24`
`pcf-services-subnet-az0`	`REGION-#a` (for example, `us-west-2a`)	`10.0.8.0/24`
`pcf-services-subnet-az1`	`REGION-#b` (for example, `us-west-2b`)	`10.0.9.0/24`
`pcf-services-subnet-az2`	`REGION-#c` (for example, `us-west-2c`)	`10.0.10.0/24`
`pcf-rds-subnet-az0`	`REGION-#a` (for example, `us-west-2a`)	`10.0.12.0/24`
`pcf-rds-subnet-az1`	`REGION-#b` (for example, `us-west-2b`)	`10.0.13.0/24`
`pcf-rds-subnet-az2`	`REGION-#c` (for example, `us-west-2c`)	`10.0.14.0/24`

See Step 4: Create a VPC.

NAT Instance

You must create a NAT instance when creating a VPC. The NAT instance must have the following configuration:

- **Instance type**: `t2.medium`
- **Key pair name**: `pcf-ops-manager-key`
- **Enable DNS hostnames**: Yes
- **Hardware tenancy**: Default

See Step 4: Create a VPC.

You must also assign the NAT instance to the `pcf-nat-security-group`. See Step 17: Secure the NAT Instance.

Security Groups

The following sections describe the security groups you must create from the EC2 Dashboard.

Ops Manager

The Ops Manager Security Group must be named `pcf-ops-manager-security-group` and have the following inbound rules:

Type	Protocol	Port Range	Source
HTTP	TCP	80	My IP
HTTPS	TCP	443	My IP
SSH	TCP	22	My IP
BOSH Agent	TCP	6868	10.0.0.0/16
BOSH Director	TCP	25555	10.0.0.0/16

See Step 5: Configure a Security Group for Ops Manager.

PCF VMs

The PCV VMs Security Group must be named `pcf-vms-security-group` and have the following inbound rule:

Type	Protocol	Port Range	Source	
All traffic	All	0 - 65535	Custom IP	10.0.0.0/16

See Step 6: Configure a Security Group for PCF VMs.

Web ELB

The Web ELB Security Group must be named `pcf-web-elb-security-group` and have the following inbound rules:

Type	Protocol	Port Range	Source	
Custom TCP rule	TCP	4443	Anywhere	0.0.0.0/0
HTTP	TCP	80	Anywhere	0.0.0.0/0
HTTPS	TCP	443	Anywhere	0.0.0.0/0

See Step 7: Configure a Security Group for the Web ELB.

SSH ELB

The SSH ELB Security Group must be named `pcf-ssh-elb-security-group` and have the following inbound rule:

Type	Protocol	Port Range	Source	

Custom TCP rule	TCP	2222	Anywhere	0.0.0.0/0

The SSH ELB Security Group must have the following outbound rule:

Type	Protocol	Port Range	Source	
All traffic	All	All	Anywhere	0.0.0.0/0

See Step 8: Configure a Security Group for the SSH ELB.

TCP ELB

The TCP ELB Security Group must be named `pcf-tcp-elb-security-group` and have the following inbound rule:

Type	Protocol	Port Range	Source	
Custom TCP rule	TCP	1024 - 1123	Anywhere	0.0.0.0/0

The TCP ELB Security Group must have the following outbound rule:

Type	Protocol	Port Range	Source	
All traffic	All	All	Anywhere	0.0.0.0/0

See Step 9: Configure a Security Group for the TCP ELB.

Outbound NAT

The Outbound NAT Security Group must be named `pcf-nat-security-group` and have the following inbound rule:

Type	Protocol	Port Range	Source	
All traffic	All	All	Custom IP	10.0.0.0/16

See Step 10: Configure a Security Group for the Outbound NAT.

MySQL

The MySQL Security Group must be named `pcf-mysql-security-group` and have the following inbound rules:

Type	Protocol	Port Range	Source	
MySQL	TCP	3306	Custom IP	10.0.0.0/16

The MySQL Security Group must have the following outbound rules:

Type	Protocol	Port Range	Destination	
All traffic	All	All	Custom IP	10.0.0.0/16

See Step 11: Configure a Security Group for MySQL.

Ops Manager AMI

You must locate the public Ops Manager AMI using the AMI ID provided by the PDF downloaded when clicking **Pivotal Cloud Foundry Ops Manager for AWS** on Pivotal Network.

See Step 12: Launch a Pivotal Ops Manager AMI.

ELBs

The following sections describe the ELBs you must create from the EC2 Dashboard.

Web ELB

You must create a web ELB with the following configuration:

- **Name**: `pcf-web-elb`
- **LB Inside**: `pcf-vpc`
- **Selected Subnet**: `pcf-public-subnet-az0` , `pcf-public-subnet-az1` , `pcf-public-subnet-az2`
- **Security Group**: `pcf-elb-security-group`
- **Health Check**: TCP Port 8080, Path: `/health`

See Step 13: Create Web Load Balancer.

SSH ELB

- **Name**: `pcf-ssh-elb`
- **LB Inside**: `pcf-vpc`
- **Selected Subnet**: `pcf-public-subnet-az0` , `pcf-public-subnet-az1` , `pcf-public-subnet-az2`
- **Security Group**: `pcf-ssh-security-group`
- **Health Check**: TCP Port 2222

See Step 14: Create SSH Load Balancer.

TCP ELB

- **Name**: `pcf-tcp-elb`
- **LB Inside**: `pcf-vpc`
- **Selected Subnet**: `pcf-public-subnet-az0` , `pcf-public-subnet-az1` , `pcf-public-subnet-az2`
- **Security Group**: `pcf-tcp-security-group`
- **Health Check**: TCP Port 80

See Step 15: Create TCP Load Balancer.

DNS Configuration

You must navigate to your DNS provider and create CNAME and A records for all three of your load balancers.

See Step 16: Configure DNS Records.

RDS Subnet Group

You must create a subnet group for RDS named `pcf-rds-subnet-group` from the RDS Dashboard.

See Step 18: Create RDS Subnet Group.

MySQL Database

You must create a MySQL database from the RDS Dashboard.

See Step 19: Create a MySQL Database using AWS RDS.

Manually Configuring Ops Manager Director for AWS

Page last updated:

This topic describes how to configure Ops Manager to deploy Pivotal Cloud Foundry (PCF) on Amazon Web Services (AWS).

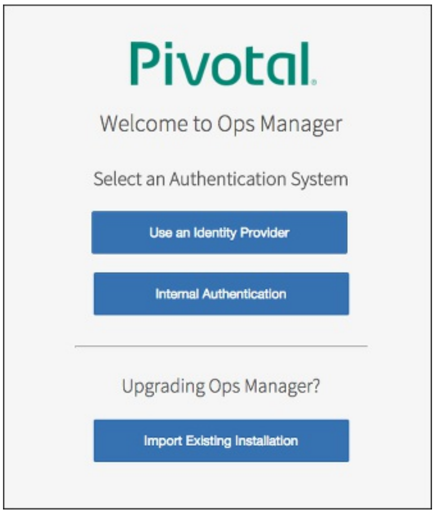
Before beginning this procedure, ensure that you have successfully completed all of the steps in the Manually Configuring AWS for PCF topic. After completing the procedures in this topic, proceed to the Manually Configuring PAS for AWS topic to continue deploying PCF.

Step 1: Access Ops Manager

> **Note**: For security, Ops Manager 1.7 and later require that you log in using a fully qualified domain name to access Ops Manager.

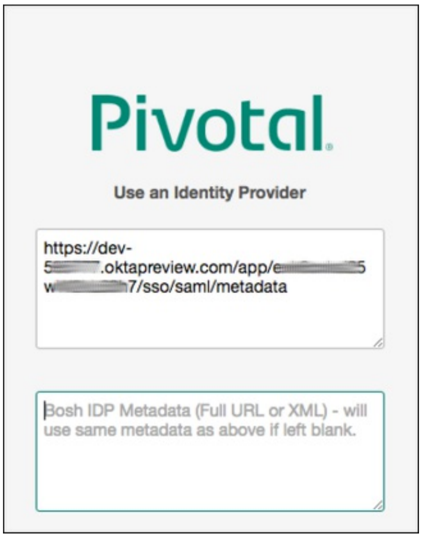
1. In a web browser, navigate to the fully qualified domain you created in the Step 16: Configure DNS Records section of the *Manually Configuring AWS for PCF* topic.

2. When Ops Manager starts for the first time, you must choose one of the following:
    - **Use an Identity Provider**: If you use an Identity Provider, an external identity server maintains your user database.
    - **Internal Authentication**: If you use Internal Authentication, PCF maintains your user database.



Use an Identity Provider (IdP)

1. Log in to your IdP console and download the IdP metadata XML. Optionally, if your IdP supports metadata URL, you can copy the metadata URL instead of the XML.



2. Copy the IdP metadata XML or URL to the Ops Manager **Use an Identity Provider** log in page.

> **Note**: The same IdP metadata URL or XML is applied for the BOSH Director. If you use a separate IdP for BOSH, copy the metadata XML or URL from that IdP and enter it into the BOSH IdP Metadata text box in the Ops Manager log in page.

3. Enter your **Decryption passphrase**. Read the **End User License Agreement**, and select the checkbox to accept the terms.

4. Your Ops Manager log in page appears. Enter your username and password. Click **Login**.

5. Download your SAML Service Provider metadata (SAML Relying Party metadata) by navigating to the following URLs:
   - **5a.** Ops Manager SAML service provider metadata: `https://OPS-MAN-FQDN:443/uaa/saml/metadata`
   - **5b.** BOSH Director SAML service provider metadata: `https://BOSH-IP-ADDRESS:8443/saml/metadata`

> **Note**: To retrieve your `BOSH-IP-ADDRESS`, navigate to the **Ops Manager Director** tile > **Status** tab. Record the **Ops Manager Director** IP address.

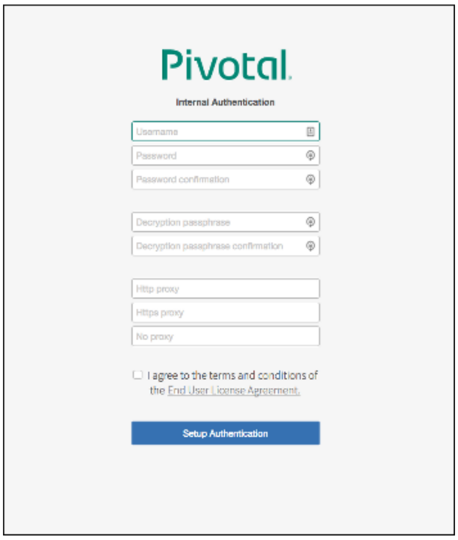
6. Configure your IdP with your SAML Service Provider metadata. Import the Ops Manager SAML provider metadata from Step 5a above to your IdP. If your IdP does not support importing, provide the values below.
   - **Single sign on** URL: `https://OPS-MAN-FQDN:443/uaa/saml/SSO/alias/OPS-MAN-FQDN`
   - **Audience URI** (SP Entity ID): `https://OP-MAN-FQDN:443/uaa`
   - **Name ID**: Email Address
   - SAML authentication requests are always signed

7. Import the BOSH Director SAML provider metadata from Step 5b to your IdP. If the IdP does not support an import, provide the values below.
   - **Single sign on** URL: `https://BOSH-IP:8443/saml/SSO/alias/BOSH-IP`
   - **Audience URI** (SP Entity ID): `https://BOSH-IP:8443`
   - **Name ID**: Email Address
   - SAML authentication requests are always signed

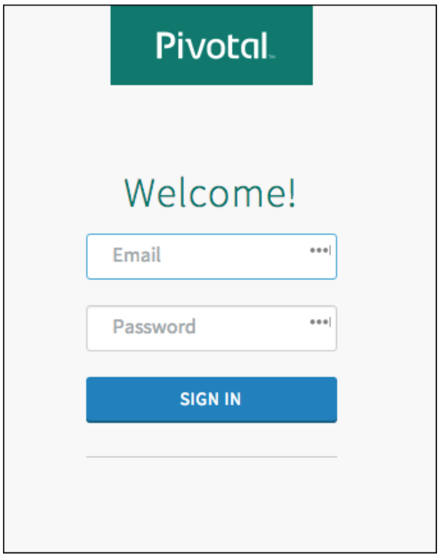
8. Return to the **Ops Manager Director** tile, and continue with the configuration steps below.

Internal Authentication

1. When redirected to the **Internal Authentication** page, you must complete the following steps:
   - Enter a **Username**, **Password**, and **Password confirmation** to create an Admin user.
   - Enter a **Decryption passphrase** and the **Decryption passphrase confirmation**. This passphrase encrypts the Ops Manager datastore, and is not recoverable.
   - If you are using an **HTTP proxy** or **HTTPS proxy**, follow the instructions in the Configuring Proxy Settings for the BOSH CPI topic.
   - Read the **End User License Agreement**, and select the checkbox to accept the terms.
   - Click **Setup Authentication**.



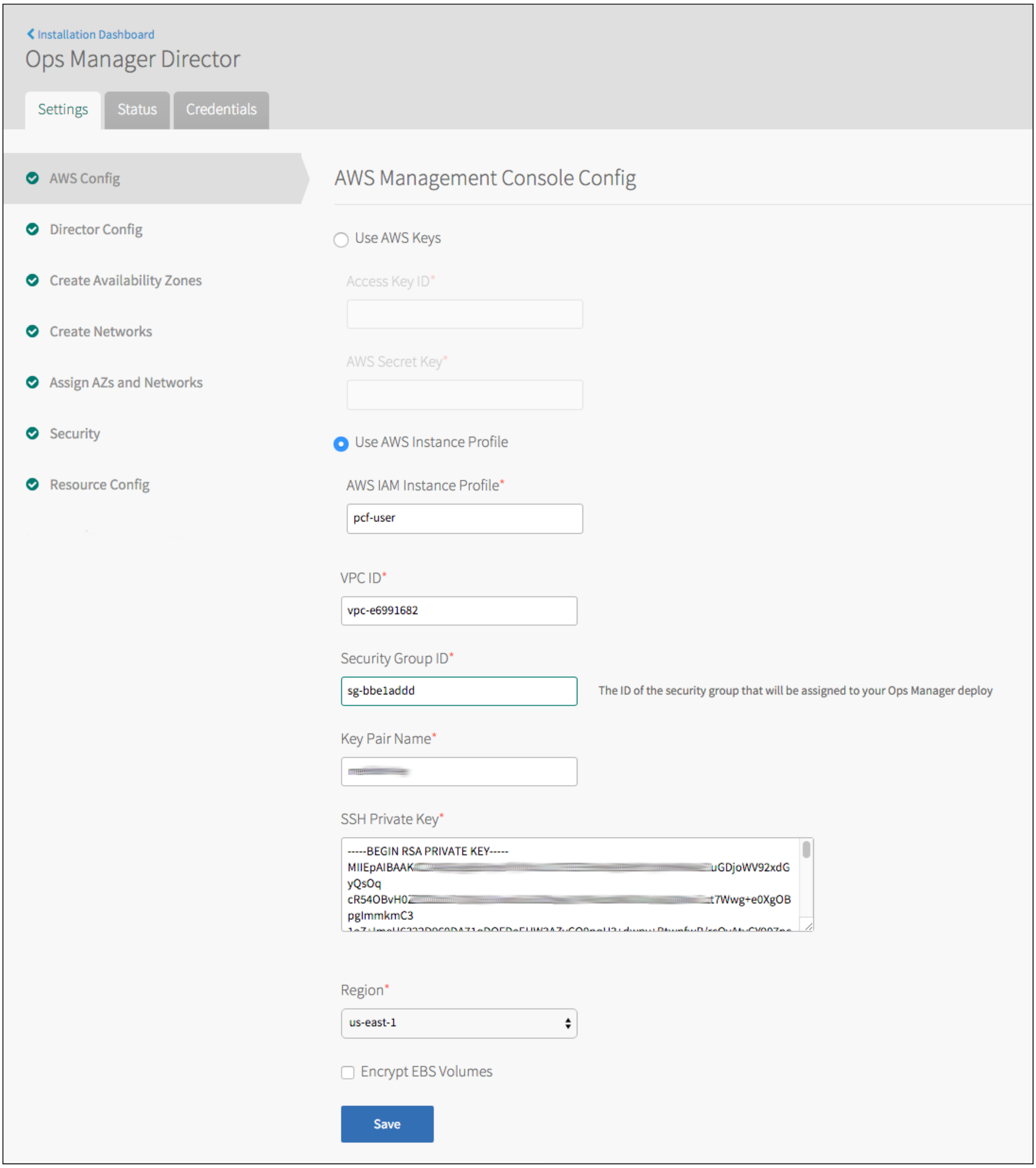
2. Log in to Ops Manager with the Admin username and password you created in the previous step.



Step 2: AWS Config Page

1. Click the **Ops Manager Director** tile.

2. Select **AWS Config**.



3. Select **Use AWS Keys** or **Use AWS Instance Profile**.
    - **Access Key ID** and **AWS Secret Key**: To retrieve your AWS key information, use the AWS Identity and Access Management (IAM) credentials that you generated in the Step 3: Create an IAM User for PCF section of the *Manually Configuring AWS for PCF* topic.
    - **AWS IAM Instance Profile**: Enter the name of the IAM profile you created in the Step 3: Create an IAM User for PCF section of the *Manually Configuring AWS for PCF* topic.

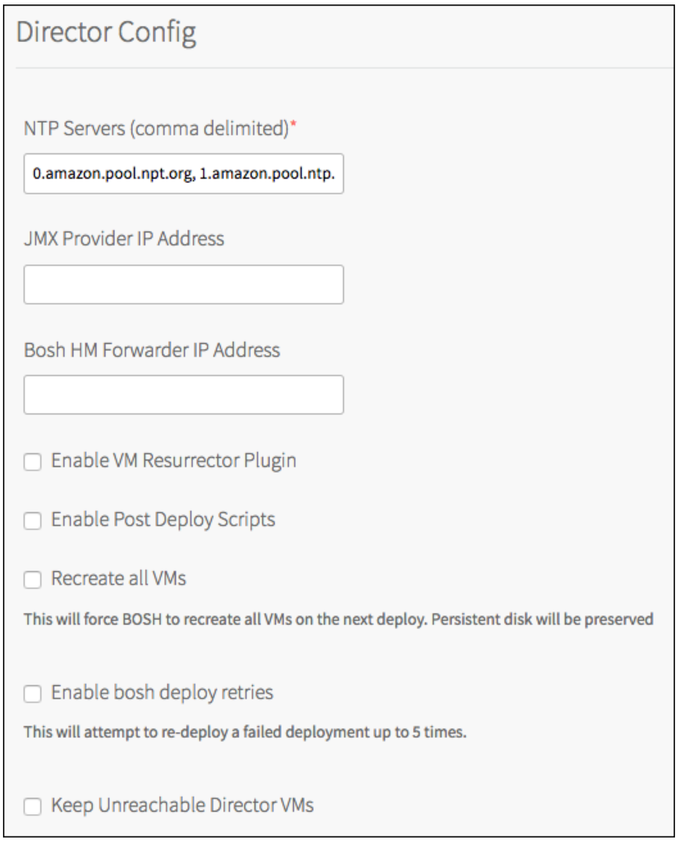
4. Complete the rest of the **AWS Management Console Config** page:
    - **VPC ID**: Enter your PCF VPC ID. Locate the VPC ID on the AWS VPC Dashboard page next to the VPC name.
    - **Security Group ID**: Enter the **Group ID** of the `pcf-vms-security-group` you created for your PCF VMs in the Step 6: Configure a Security Group for PCF VMs section of the *Manually Configuring AWS for PCF* topic. Locate the Group ID in the **Security Groups** tab of your EC2 Dashboard.
    - **Key Pair Name**: Enter `pcf-ops-manager-key`.
    - **SSH Private Key**: Open the AWS key pair `pcf-ops-manager-keys.pem` file you generated in the Step 3: Create an IAM User for PCF section of the *Manually Configuring AWS for PCF* topic. Copy the contents of the `.pem` file and paste it into the **SSH Private Key** field.
    - **Region**: Select the region where you deployed Ops Manager.
    - **Encrypt EBS Volumes**: Select this checkbox to enable full encryption on persistent disks of all BOSH-deployed VMs except the Ops Manager VM

and Director VM. See the Configuring Amazon EBS Encryption for PCF on AWS topic for details about using EBS encryption.

5. Click **Save**.

Step 3: Director Config Page

1. Select **Director Config**.



2. Enter at least two of the following NTP servers in the **NTP Servers (comma delimited)** field, separated by a comma: `0.amazon.pool.ntp.org,1.amazon.pool.ntp.org,2.amazon.pool.ntp.org,3.amazon.pool.ntp.org`

3. Leave the **JMX Provider IP Address** field blank.

> **Note**: Starting from PCF v2.0, BOSH-reported system metrics are available in the Loggregator Firehose by default. Therefore, if you continue to use PCF JMX Bridge for consuming them outside of the Firehose, you may receive duplicate data. To prevent this, leave the **JMX Provider IP Address** field blank. For additional guidance, see BOSH System Metrics Available in Loggregator Firehose.

4. Leave the **Bosh HM Forwarder IP Address** field blank.

> **Note**: Starting from PCF v2.0, BOSH-reported system metrics are available in the Loggregator Firehose by default. Therefore, if you continue to use the BOSH HM Forwarder for consuming them, you may receive duplicate data. To prevent this, leave the **Bosh HM Forwarder IP Address** field blank. For additional guidance, see BOSH System Metrics Available in Loggregator Firehose.

5. Select the **Enable VM Resurrector Plugin** checkbox to enable the Ops Manager Resurrector functionality and increase Pivotal Application Service (PAS) availability. For more information, see the Using Ops Manager Resurrector on VMware vSphere topic.

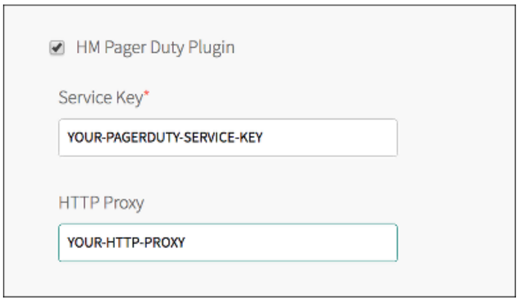
6. Select **Enable Post Deploy Scripts** to run a post-deploy script after deployment. This script allows the job to execute additional commands against a deployment.

7. Select **Recreate all VMs** to force BOSH to recreate all VMs on the next deploy. This process does not destroy any persistent disk data.

8. Select **Enable bosh deploy retries** if you want Ops Manager to retry failed BOSH operations up to five times.

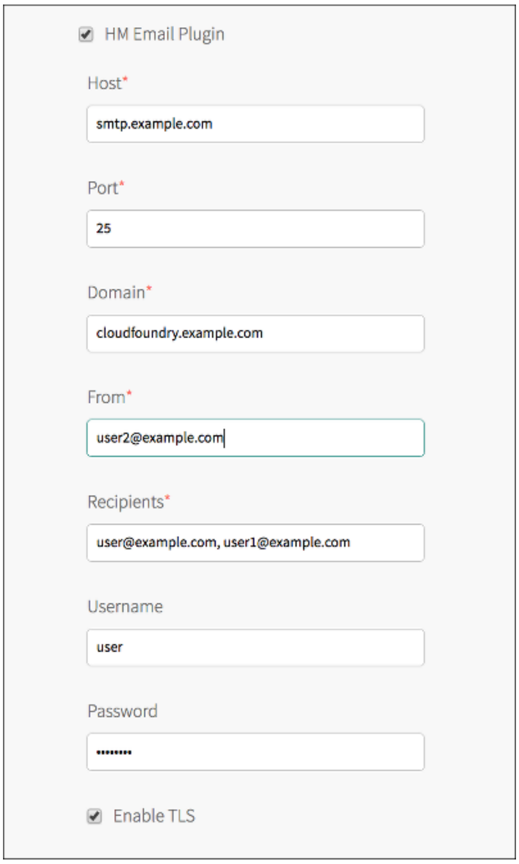
9. Select **Keep Unreachable Director VMs** if you want to preserve Ops Manager Director VMs after a failed deployment for troubleshooting purposes.

10. (Optional) Select **HM Pager Duty Plugin** to enable Health Monitor integration with PagerDuty.



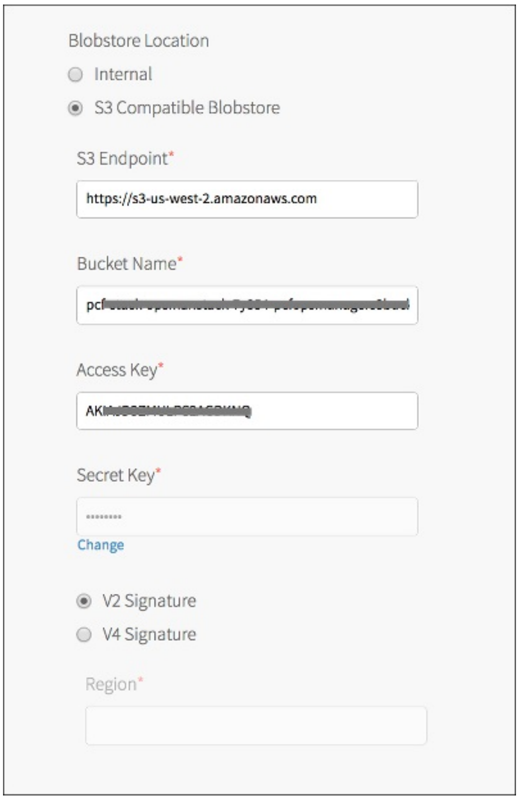
    - **Service Key**: Enter your API service key from PagerDuty.
    - **HTTP Proxy**: Enter an HTTP proxy for use with PagerDuty.

11. (Optional) Select **HM Email Plugin** to enable Health Monitor integration with email.



    - **Host**: Enter your email hostname.
    - **Port**: Enter your email port number.
    - **Domain**: Enter your domain.
    - **From**: Enter the address for the sender.
    - **Recipients**: Enter comma separated addresses of intended recipients.
    - **Username**: Enter the username for your email server.
    - **Password**: Enter your username's password.
    - **Enable TLS**: Select this checkbox to enable Transport Layer Security.

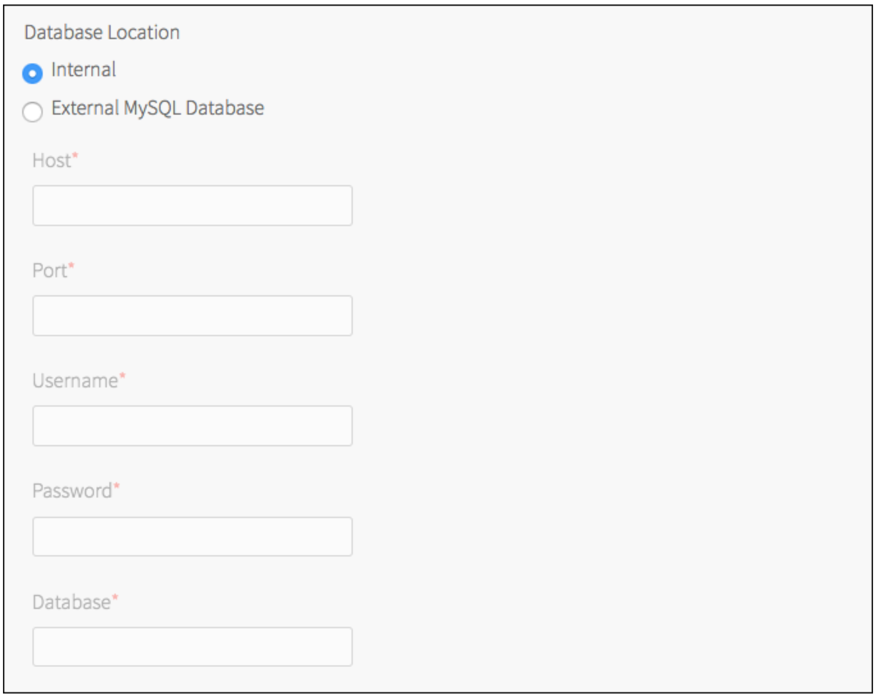
12. For **Blobstore Location**, select **S3 Compatible Blobstore** and complete the following steps:



- **S3 Endpoint**: Navigate to the [Regions and Endpoints](#) topic in the AWS documentation. Locate the endpoint for your region in the **Amazon Simple Storage Service (S3)** table and construct a URL using your region's endpoint. For example, if you are using the `us-west-2` region, the URL you create would be `https://s3-us-west-2.amazonaws.com`. Enter this URL into the **S3 Endpoint** field in Ops Manager.
- **Bucket Name**: Enter the Ops Manager bucket name that you defined in the [Manually Configuring PCF for AWS](#) topic.
- **Access Key** and **Secret Key**: To retrieve your AWS key information, use the IAM credentials that you generated in the [Manually Configuring AWS for PCF](#) topic.
- Select **V2 Signature** or **V4 Signature**. If you select **V4 Signature**, enter your **Region**.

> **Note**: AWS recommends using Signature Version 4. For more information about AWS S3 Signatures, see the [Authenticating Requests](#) documentation.

13. For **Database Location**, select **External MySQL Database** and complete the following steps:



- From the AWS Console, navigate to the RDS Dashboard.

- Select **Instances**, then click the arrow to the left of your instance and select the second icon to display the **Details** information.
- Refer to the following table to retrieve the values for the **Director Config** page:

RDS Instance Field	Ops Manager Director Field
Endpoint	**Host**
Port	**Port**, which is `3306`.
DB Name	**Database**, which is `bosh`.
Username	**Username**

- For **Password**, enter the password that you defined for your MySQL database when you created in the Step 19: Create a MySQL Database using AWS RDS section of the *Manually Configuring AWS for PCF* topic.

14. (Optional) **Director Workers** sets the number of workers available to execute Director tasks. This field defaults to 5.

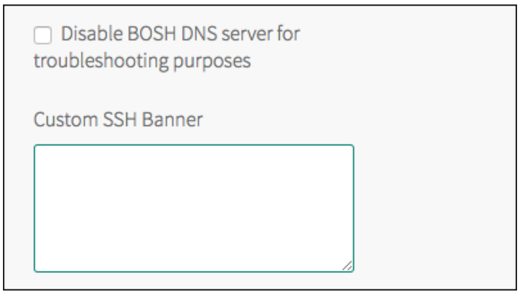
15. (Optional) **Max Threads** sets the maximum number of threads that the Ops Manager Director can run simultaneously. For AWS, the default value is 6. Leave this field blank to use this default value. Pivotal recommends that you use the default value unless doing so results in rate limiting or errors on your IaaS.

16. (Optional) To add a custom URL for your Ops Manager Director, enter a valid hostname in **Director Hostname**. You can also use this field to configure a load balancer in front of your Ops Manager Director.

17. (Optional) To disable BOSH DNS, select the **Disable BOSH DNS server for troubleshooting purposes** checkbox. For more information about the BOSH DNS service discovery mechanism, see BOSH DNS Service Discovery for Application Containers in the Pivotal Application Service (PAS) Release Notes.

Breaking Change: Do not disable BOSH DNS without consulting Pivotal Support. For more information about disabling BOSH DNS, see Disabling or Opting Out of BOSH DNS in PCF in the Pivotal Knowledge Base.

18. (Optional) To set a custom banner that users see when logging in to the Director using SSH, enter text in the **Custom SSH Banner** field.

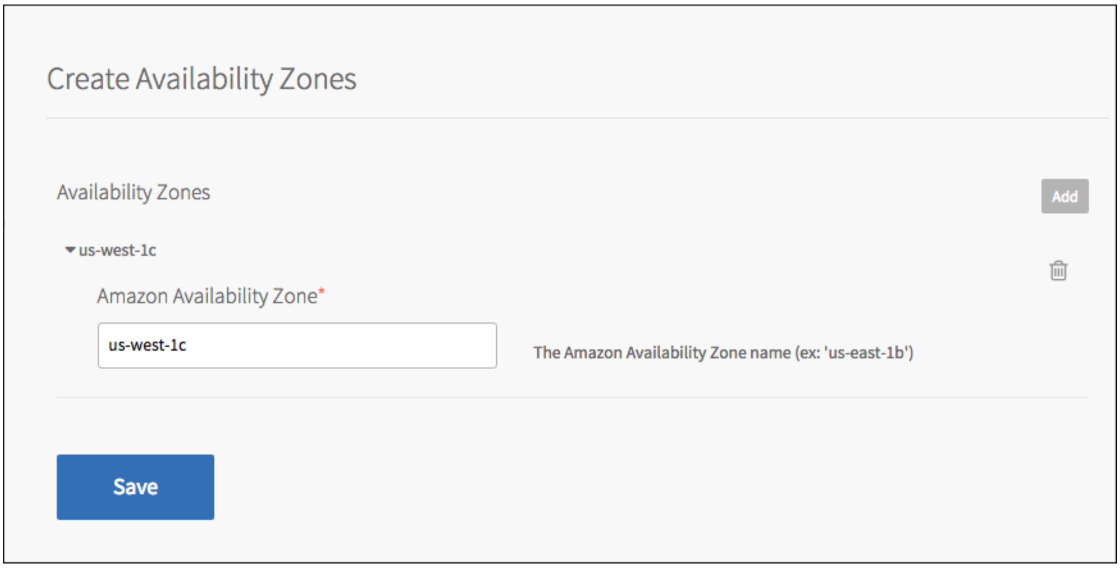


19. Click **Save**.

Step 4: Create Availability Zones Page

Note: Pivotal recommends at least three availability zones (AZs) for a highly available installation of PAS. The procedures in Manually Configuring AWS for PCF use 3 AZs.

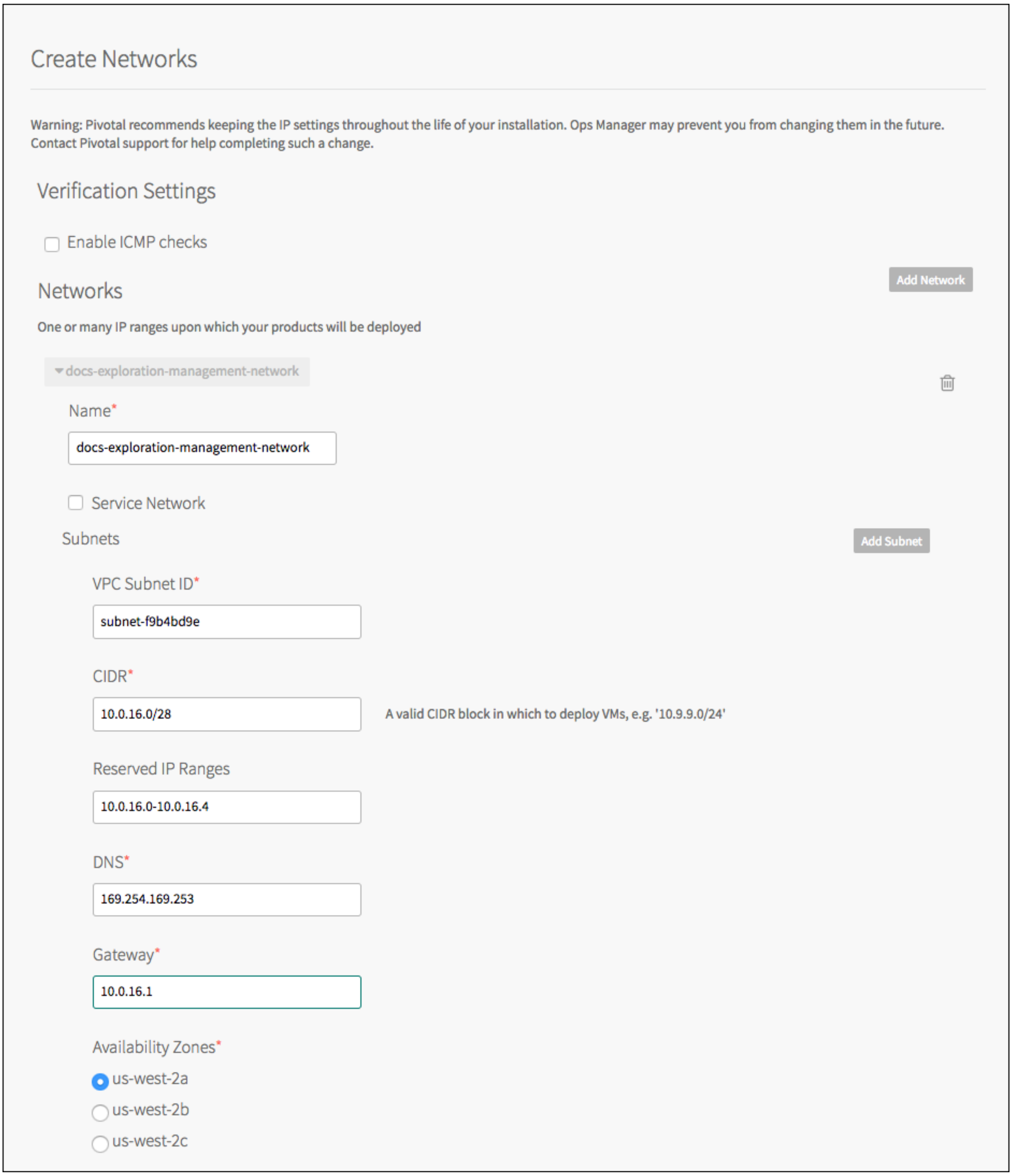
1. Select **Create Availability Zones**.



2. Perform the following steps to add the three AZs you specified in the Step 4: Create a VPC section of the *Manually Configuring AWS for PCF* topic:
    a. Click **Add**.
    b. For **Amazon Availability Zone**, enter the name of the AZ.
    c. Repeat until you have entered all three AZs, in the format `REGION-#a`, `REGION-#b`, and `REGION-#c`. For example, `us-west-2a`, `us-west-2b`, and `us-west-2c`.
3. Click **Save**.

Step 5: Create Networks Page

1. Select **Create Networks**.



2. (Optional) Select **Enable ICMP checks** to enable ICMP on your networks. Ops Manager uses ICMP checks to confirm that components within your network are reachable.

3. Perform the following steps to add the network configuration you created for your VPC in the Step 4: Create a VPC section of the *Manually Configuring PCF for AWS* topic:

   a. Click **Add Network**.
   b. For **Name**, enter `pcf-management-network`.
   c. Create a subnet for each availability zone by clicking **Add Subnet**. Refer to the table below for the information required to create all three subnets:

VPC Subnet ID	CIDR	Reserved IP Ranges	DNS	Gateway	Availability Zones
`pcf-management-subnet-az0`	`10.0.16.0/28`	`10.0.16.0-10.0.16.4`	`169.254.169.253`	`10.0.16.1`	`REGION-#a` (for example, `us-west-2a`)
`pcf-management-subnet-az1`	`10.0.16.16/28`	`10.0.16.16-10.0.16.20`	`169.254.169.253`	`10.0.16.17`	`REGION-#b` (for example, `us-west-2b`)

`pcf-management-subnet-az2`	`10.0.16.32/28`	`10.0.16.32-10.0.16.36`	`169.254.169.253`	`10.0.16.33`	`REGION-#c` (for example, `us-west-2c`)

d. Click **Add Network**.
e. For **Name**, enter `pcf-ert-network`.
f. Create a subnet for each availability zone by clicking **Add Subnet**. Refer to the table below for the information required to create all three subnets:

VPC Subnet ID	CIDR	Reserved IP Ranges	DNS	Gateway	Availability Zones
`pcf-ert-subnet-az0`	`10.0.4.0/24`	`10.0.4.0-10.0.4.4`	`169.254.169.253`	`10.0.4.1`	`REGION-#a` (for example, `us-west-2a`)
`pcf-ert-subnet-az1`	`10.0.5.0/24`	`10.0.5.0-10.0.5.4`	`169.254.169.253`	`10.0.5.1`	`REGION-#b` (for example, `us-west-2b`)
`pcf-ert-subnet-az2`	`10.0.6.0/24`	`10.0.6.0-10.0.6.4`	`169.254.169.253`	`10.0.6.1`	`REGION-#c` (for example, `us-west-2c`)

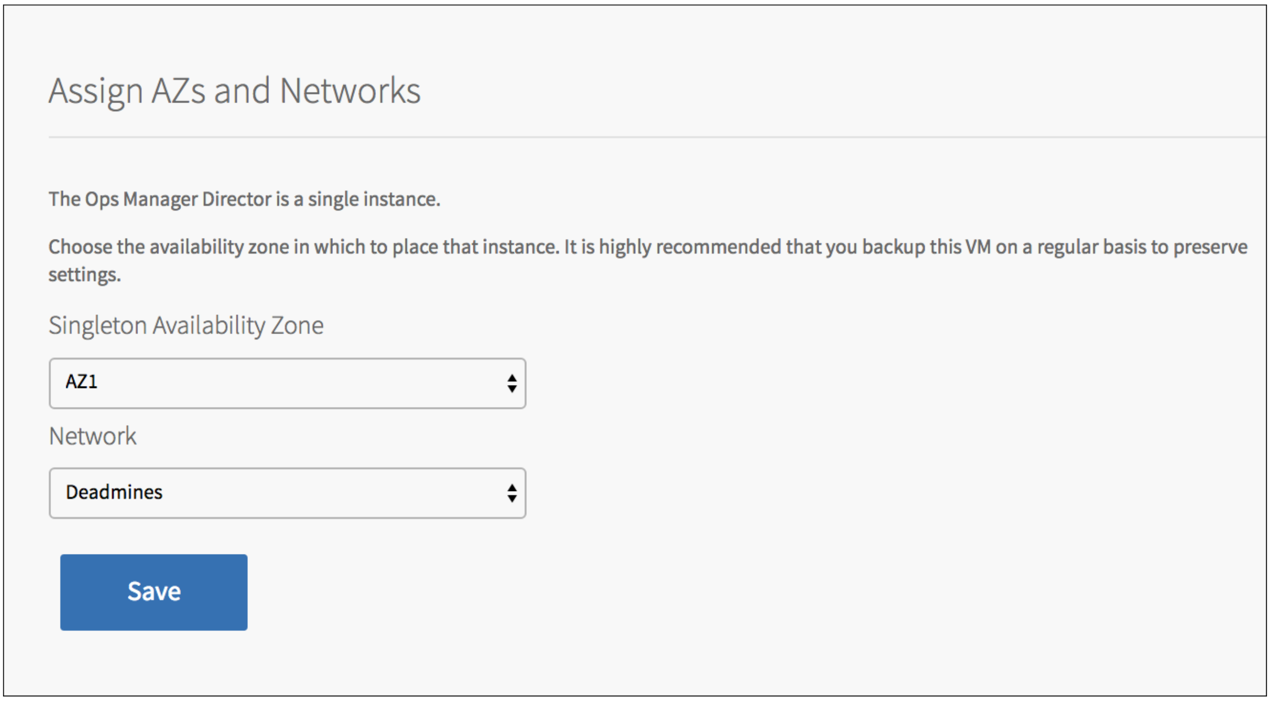
g. Click **Add Network**.
h. Check the **Service Network** box. This allows you to dynamically provision VMs in this network for use with on-demand services. Ops Manager does not provision VMs within the specified CIDR range.
i. Enter `pcf-services-network` in **Name** field.
j. Create a subnet for each availability zone by clicking **Add Subnet**. Refer to the table below for the information required to create all three subnets:

VPC Subnet ID	CIDR	Reserved IP Ranges	DNS	Gateway	Availability Zones
`pcf-services-subnet-az0`	`10.0.8.0/24`	`10.0.8.0-10.0.8.3`	`169.254.169.253`	`10.0.8.1`	`REGION-#a` (for example, `us-west-2a`)
`pcf-services-subnet-az1`	`10.0.9.0/24`	`10.0.9.0-10.0.9.3`	`169.254.169.253`	`10.0.9.1`	`REGION-#b` (for example, `us-west-2b`)
`pcf-services-subnet-az2`	`10.0.10.0/24`	`10.0.10.0-10.0.10.3`	`169.254.169.253`	`10.0.10.1`	`REGION-#c` (for example, `us-west-2c`)

4. Click **Save**.

Step 6: Assign AZs and Networks

1. Select **Assign AZs and Networks**.



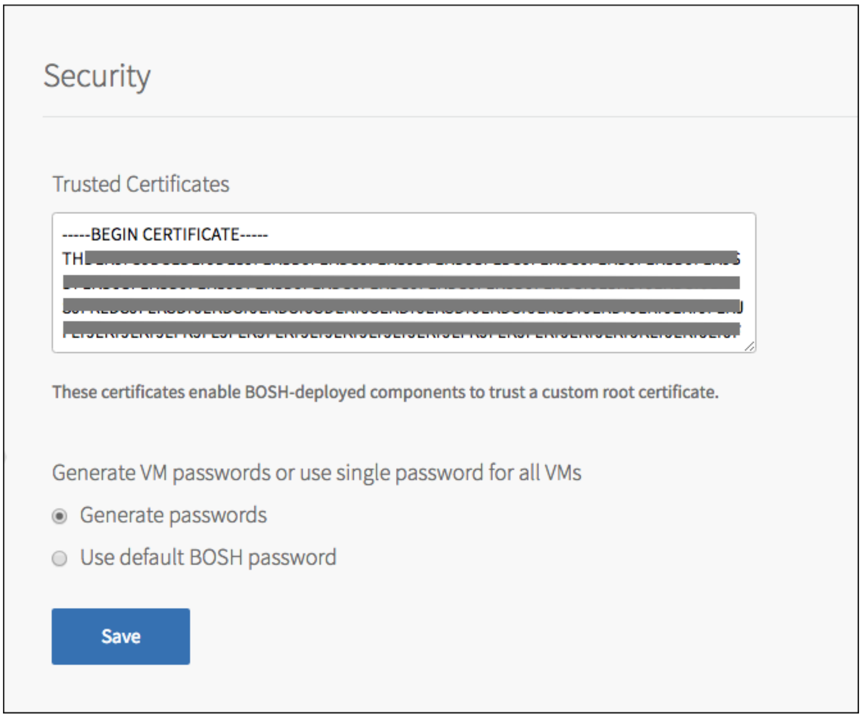
2. Use the drop-down menu to select a **Singleton Availability Zone**. The Ops Manager Director is deployed into this AZ.

3. Use the drop-down menu to select `pcf-management-network` under **Network**. The Ops Manager Director is deployed into this network.

4. Click **Save**.

Step 7: Security Page

1. Select **Security**.



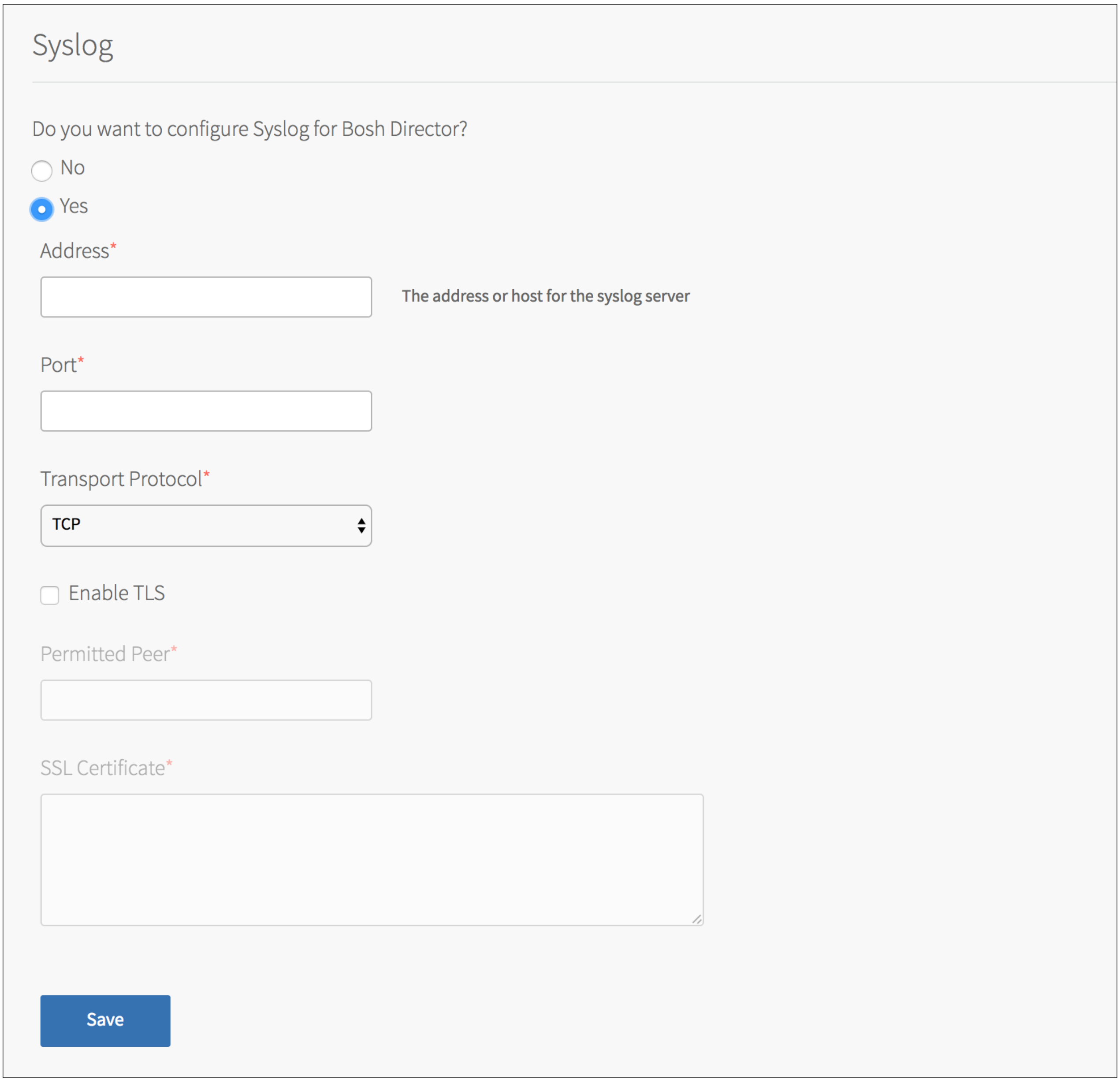
2. In **Trusted Certificates**, enter a custom certificate authority (CA) certificate to insert into your organization's certificate trust chain. This feature enables all BOSH-deployed components in your deployment to trust a custom root certificate. If you want to use Docker Registries for running app instances in Docker containers, use this field to enter your certificate for your private Docker Registry. See the Using Docker Registries topic for more information.

3. Choose **Generate passwords** or **Use default BOSH password**. Pivotal recommends that you use the **Generate passwords** option for greater security.

4. Click **Save**. To view your saved Director password, click the **Credentials** tab.

Step 8: Syslog Page

1. Select **Syslog**.



2. (Optional) To send BOSH Director system logs to a remote server, select **Yes**.

3. In the **Address** field, enter the IP address or DNS name for the remote server.

4. In the **Port** field, enter the port number that the remote server listens on.

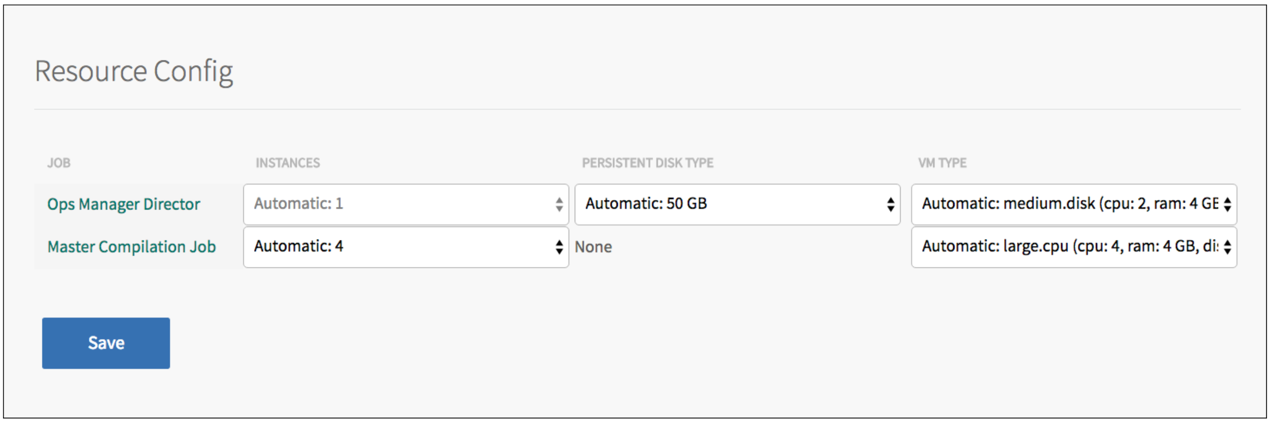
5. In the **Transport Protocol** dropdown menu, select **TCP**, **UDP**, or **RELP**. This selection determines which transport protocol is used to send the logs to the remote server.

6. (Optional) Mark the **Enable TLS** checkbox to use TLS encryption when sending logs to the remote server.
   - In the **Permitted Peer** field, enter either the name or SHA1 fingerprint of the remote peer.
   - In the **SSL Certificate** field, enter the SSL certificate for the remote server.

7. Click **Save**.

Step 9: Resource Config Page

1. Select **Resource Config**.



2. Adjust any values as necessary for your deployment. Under the **Instances**, **Persistent Disk Type**, and **VM Type** fields, choose **Automatic** from the drop-down menu to allocate the recommended resources for the job. If the **Persistent Disk Type** field reads **None**, the job does not require persistent disk space.

> **Note**: Pivotal recommends provisioning an Ops Manager Director VM with at least 8 GB memory.

> **Note**: If you set a field to **Automatic** and the recommended resource allocation changes in a future version, Ops Manager automatically uses the new recommended allocation.

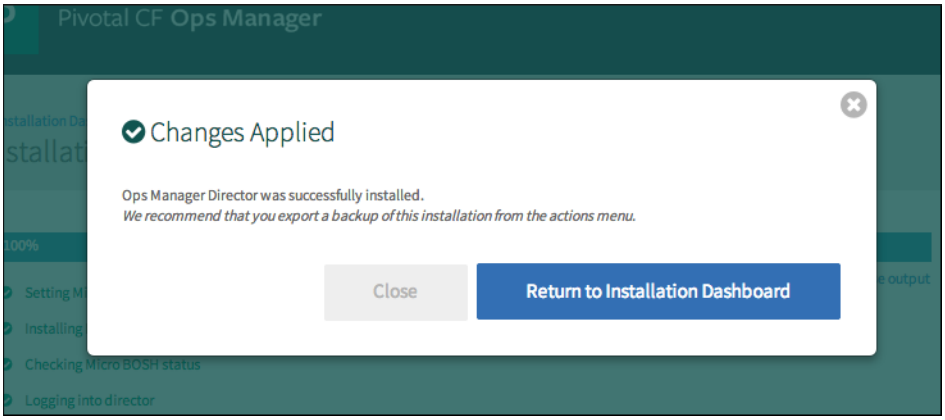
3. Click **Save**.

Step 10: Complete the Ops Manager Director Installation

1. Return to the **Installation Dashboard**.

2. Click **Apply Changes**.

3. Ops Manager Director begins to install. The **Changes Applied** window displays when the installation process successfully completes.



4. Proceed to the Manually Configuring PAS for AWS topic to continue deploying PCF.

Manually Configuring PAS for AWS

This topic describes how to configure Pivotal Application Service (PAS) components as part of deploying Pivotal Cloud Foundry (PCF) on Amazon Web Services (AWS).

Before beginning this procedure, ensure that you have successfully completed all steps in the Manually Configuring AWS for PCF and Configuring Ops Manager Director for AWS.

Note: If you plan to install the PCF IPsec add-on, you must do so before installing any other tiles. Pivotal recommends installing IPsec immediately after Ops Manager, and before installing the PAS Runtime tile.

Step 1: Add PAS to Ops Manager

1. If you have not already downloaded PAS, log in to Pivotal Network, and click on PAS

2. From the **Releases** drop-down, select the release to install and choose one of the following:

    a. Click PAS download the PAS `.pivotal` file
    b. Click **PCF Small Footprint Runtime** to download the Small Footprint Runtime `.pivotal` file. For more information, see Getting Started with Small Footprint Runtime.

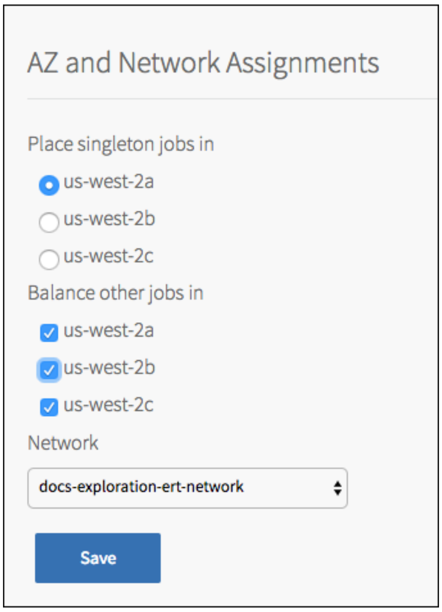
3. Navigate to the Pivotal Cloud Foundry Operations Manager Installation Dashboard.

4. Click **Import a Product** to add your tile to Ops Manager. For more information, refer to the Adding and Deleting Products topic.

5. Click the PAS tile in the Installation Dashboard.

Step 2: Assign Availability Zones and Networks

1. Select **Assign AZs and Networks**. These are the Availability Zones that you create when configuring Ops Manager Director.



2. Select an Availability Zone under **Place singleton jobs**. Ops Manager runs any job with a single instance in this Availability Zone.

3. Select all Availability Zones under **Balance other jobs**. Ops Manager balances instances of jobs with more than one instance across the Availability Zones that you specify.

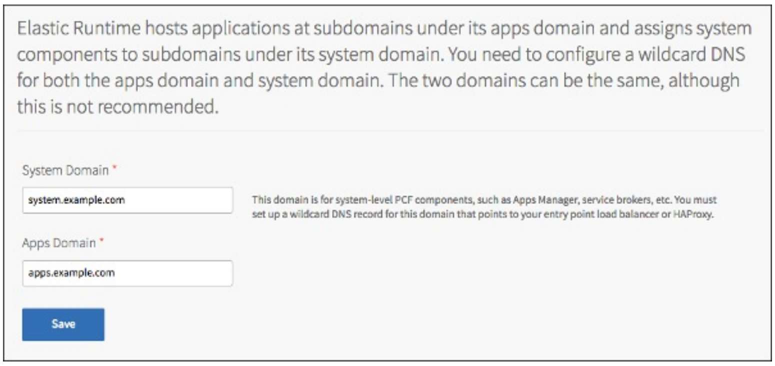
> **Note:** Pivotal recommends at least three Availability Zones for a highly available installation of PAS.

4. For **Network**, select the `pcf-ert-network` network that you created in the Step 5: Create Networks Page section of the *Manually Configuring Ops Manager Director for AWS* topic.

5. Click **Save**.

Step 3: Configure Domains

1. Select **Domains**.



2. Enter the system and application domains.
   - The **System Domain** defines your target when you push apps to PAS. This the `system.YOUR-SYSTEM-DOMAIN.com` domain that you created in Manually Configuring AWS for PCF.
   - The **Apps Domain** defines where PAS should serve your apps. This the `apps.YOUR-SYSTEM-DOMAIN.com` domain that you created in Manually Configuring AWS for PCF.

3. Click **Save**.

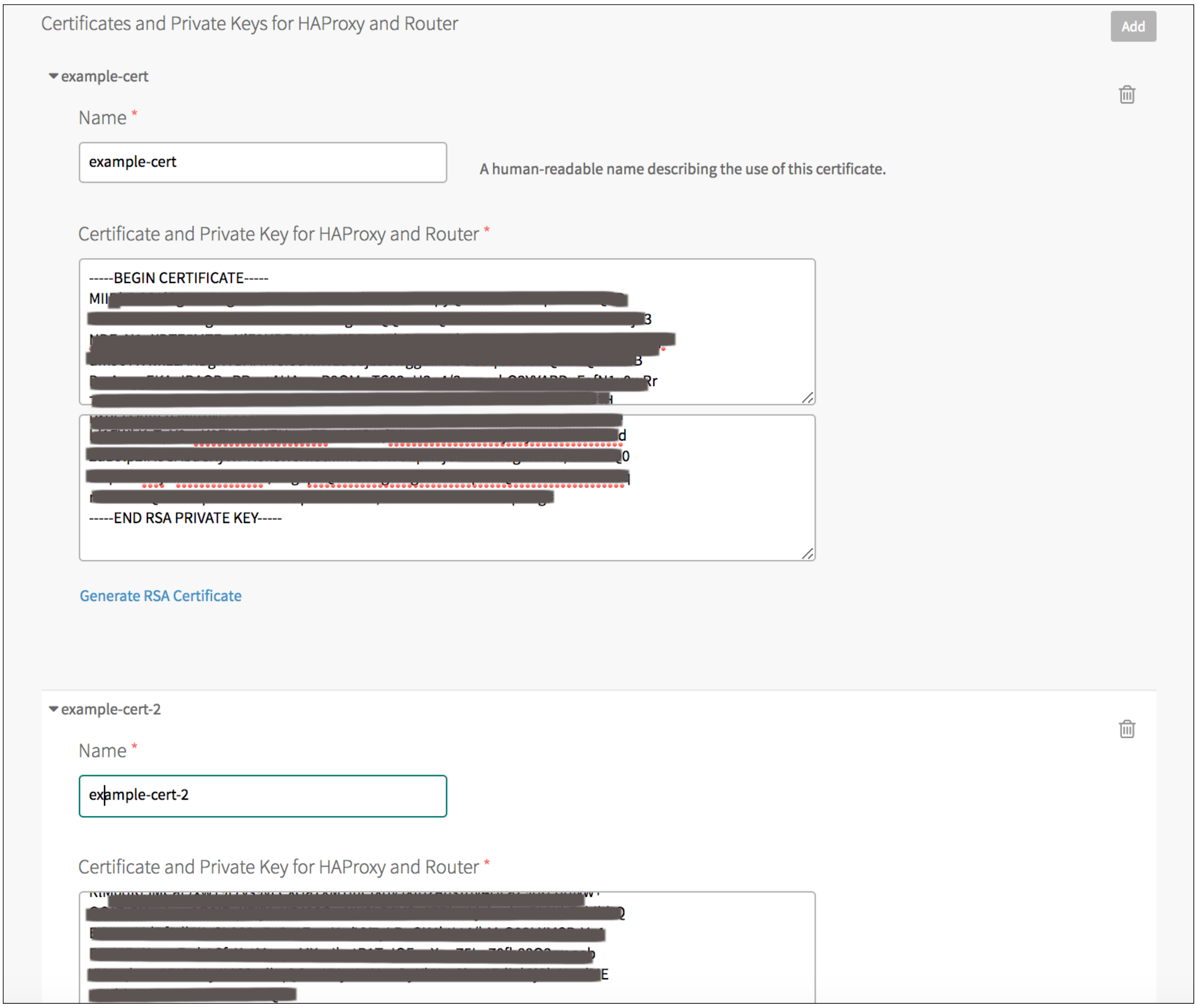
Step 4: Configure Networking

1. Select **Networking**.

2. Leave the **Router IPs**, **SSH Proxy IPs**, **HAProxy IPs**, and **TCP Router IPs** fields blank. You do not need to complete these fields when deploying PCF to AWS with Elastic Load Balancers (ELBs).

> **Note**: You specify load balancers in the **Resource Config** section of Pivotal Application Service (PAS) later in the installation process. See the Configure Router to Elastic Load Balancer section of this topic for more information.

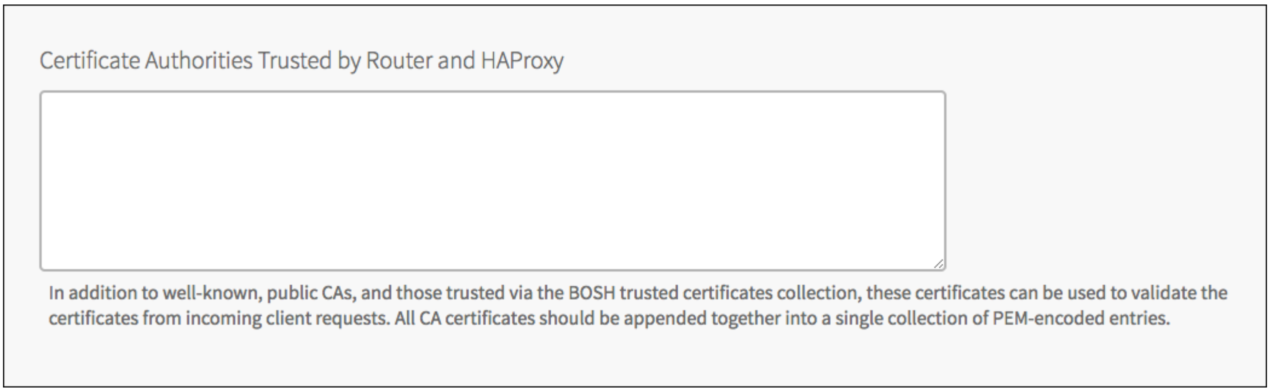
3. Under **Certificates and Private Key for HAProxy and Router**, you must provide at least one **Certificate and Private Key** name and certificate keypair for HAProxy and Gorouter. The HAProxy and Gorouter are enabled to receive TLS communication by default. You can configure multiple certificates for HAProxy and Gorouter. 1. Click the **Add** button to add a name for the certificate chain and its private keypair. This certificate is the default used by Gorouter and HAProxy.



You can either provide a certificate signed by a Certificate Authority (CA) or click on the **Generate RSA Certificate** link to generate a self-signed certificate in Ops Manager. 1. If you want to configure multiple certificates for HAProxy and Gorouter, click the **Add** button and fill in the appropriate fields for each additional certificate keypair.

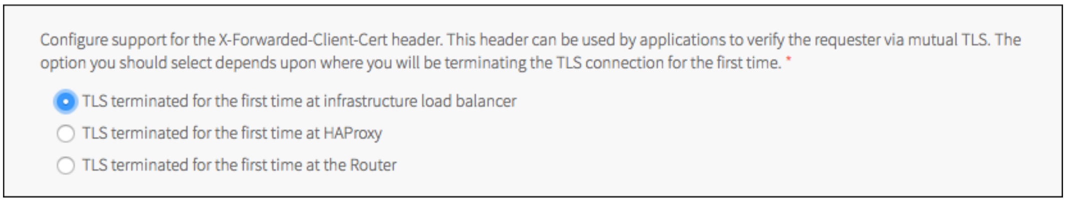
For details about generating certificates in Ops Manager for your wildcard system domains, see the Providing a Certificate for Your SSL/TLS Termination Point topic.

4. (Optional) When validating client requests using mutual TLS, the Gorouter trusts multiple certificate authorities (CAs) by default. If you want to configure the Gorouter and HAProxy to trust additional CAs, enter your CA certificates under **Certificate Authorities Trusted by Router and HAProxy**. All CA certificates should be appended together into a single collection of PEM-encoded entries.



5. In the **Minimum version of TLS supported by HAProxy and Router** field, select the minimum version of TLS to use in HAProxy and Router communications. HAProxy and Router use TLS v1.2 by default. If you need to accommodate clients that use an older version of TLS, select a lower minimum version. For a list of TLS ciphers supported by the Gorouter, see [Securing Traffic into Cloud Foundry](#).

6. Under **Configure support for the X-Forwarded-Client-Cert header**, configure PCF handles `x-forwarded-client-cert` (XFCC) HTTP headers based on where TLS is terminated for the first time in your deployment.



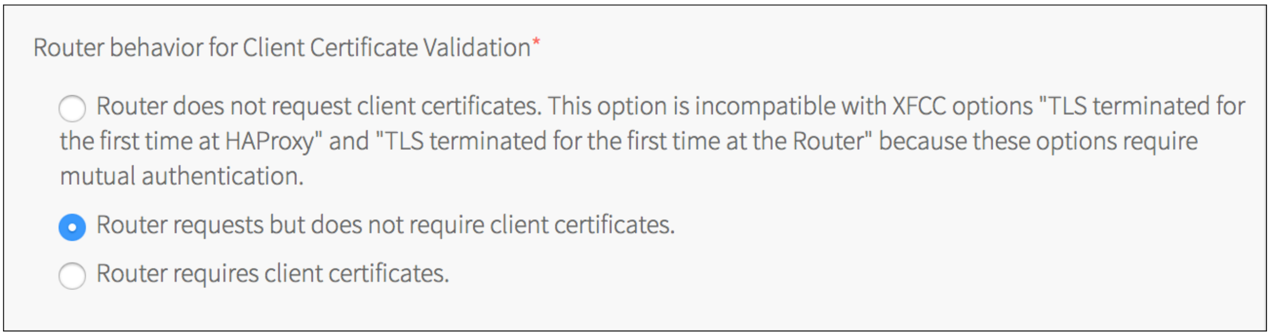
The following table indicates which option to choose based on your deployment layout.

If your deployment is configured as follows:	Then select the following option:	Additional notes:
◦ The Load Balancer is terminating TLS, and ◦ Load balancer is configured to put the client certificate from a mutual authentication TLS handshake into the X-Forwarded-Client-Cert HTTP header	TLS terminated for the first time at infrastructure load balancer (default).	Both HAProxy and the Gorouter forward the XFCC header when included in the request.
◦ The Load Balancer is configured to pass through the TLS handshake via TCP to the instances of HAProxy, and ◦ HAProxy instance count is > 0	TLS terminated for the first time at HAProxy.	HAProxy sets the XFCC header with the client certificate received in the TLS handshake. The Gorouter forwards the header. **Breaking Change**: In the **Router behavior for Client Certificates** field, you cannot select the **Router does not request client certificates** option.
◦ The Load Balancer is configured to pass through the TLS handshake via TCP to instances of the Gorouter	TLS terminated for the first time at the Gorouter.	The Gorouter strips the XFCC header if it is included in the request and forwards the client certificate received in the TLS handshake in a new XFCC header. If you have deployed instances of HAProxy, app traffic bypasses those instances in this configuration. If you have also configured your load balancer to route requests for ssh directly to the Diego Brain, consider reducing HAProxy instances to 0.

		Breaking Change: In the **Router behavior for Client Certificates** field, you cannot select the **Router does not request client certificates** option.

For a description of the behavior of each configuration option, see Forward Client Certificate to Applications.

7. To configure Gorouter behavior for handling client certificates, select one of the following options in the **Router behavior for Client Certificate Validation** field.



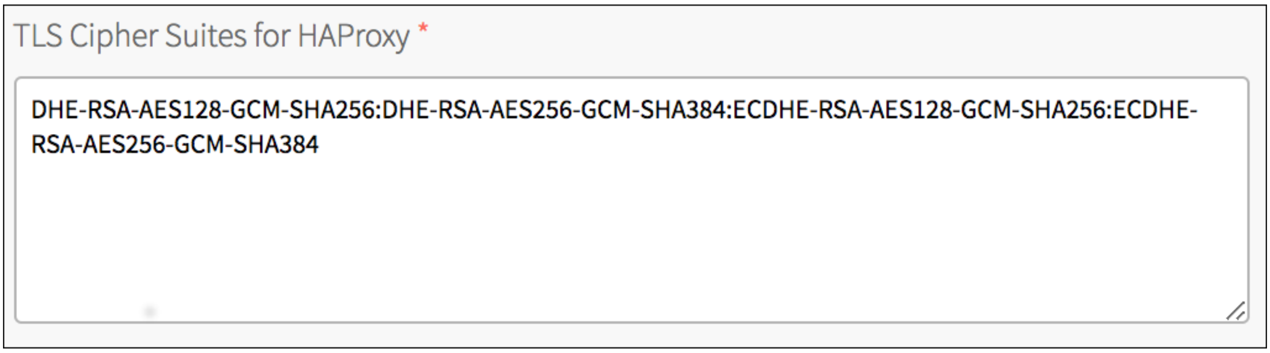
   - **Router does not request client certificates.** This option is incompatible with the XFCC configuration options **TLS terminated for the first time at HAProxy** and **TLS terminated for the first time at the Router** in PAS because these options require mutual authentication. As client certificates are not requested, client will not provide them, and thus validation of client certificates will not occur.
   - **Router requests but does not require client certificates.** Gorouter requests client certificates in TLS handshakes, validates them when presented, but does not require them. This is the default configuration.
   - **Router requires client certificates.** Gorouter validates that the client certificate is signed by a Certificate Authority that Gorouter trusts. If Gorouter cannot validate the client certificate, the TLS handshake fails.

8. In the **TLS Cipher Suites for Router** field, specify the TLS cipher suites to use for TLS handshakes between the Gorouter and downstream clients like load balancers or HAProxy. Use an ordered, colon-delimited list of Golang-supported TLS cipher suites in the OpenSSL format. The recommended setting is `ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384`. Operators should verify that the ciphers are supported by any clients or downstream components that will initiate TLS handshakes with the Gorouter. For a list of TLS ciphers supported by the Gorouter, see Securing Traffic into Cloud Foundry.

   Verify that whatever client is participating in the TLS handshake with the Gorouter has at least one cipher suite in common with the Gorouter.

   Note: Specify cipher suites that are supported by the versions configured in the **Minimum version of TLS supported by HAProxy and Router** field.

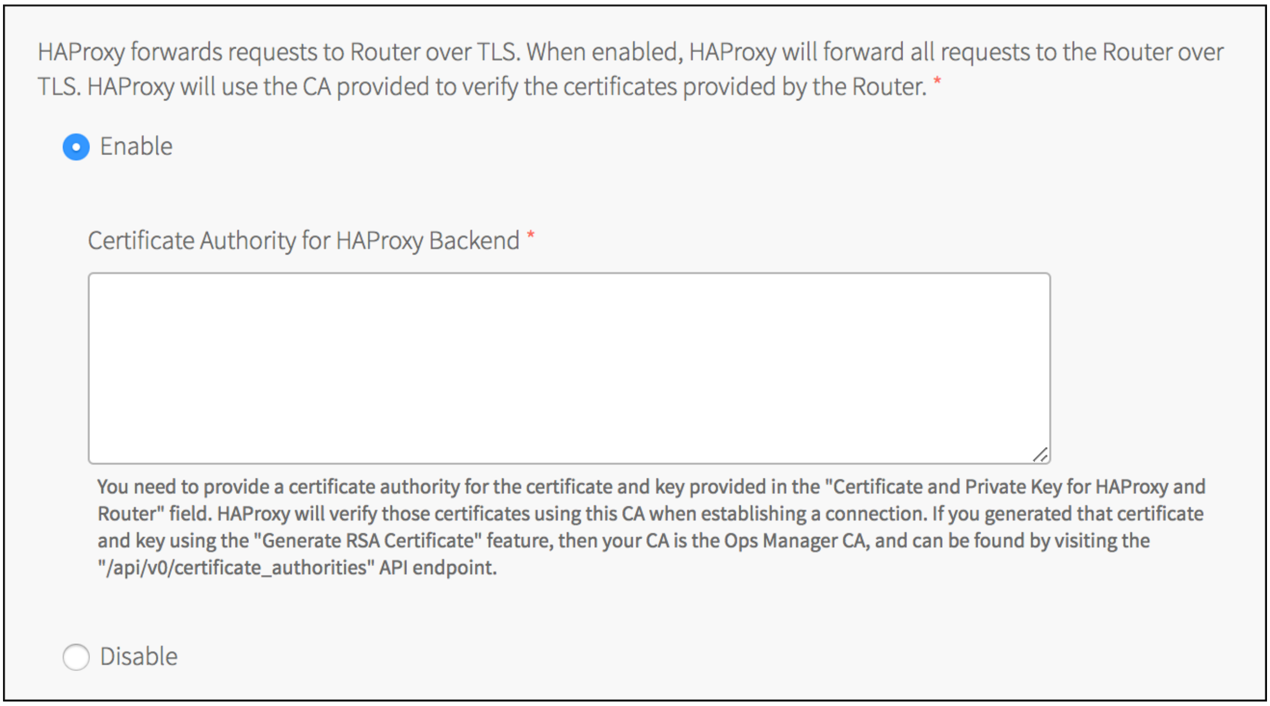
9. In the **TLS Cipher Suites for HAProxy** field, specify the TLS cipher suites to use in HAProxy for TLS handshakes between HAProxy and its clients such as load balancers and Gorouter. Use an ordered, colon-delimited list of TLS cipher suites in the OpenSSL format. The recommended setting: `DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384`
   Operators should verify that the ciphers are supported by any clients or downstream components that will initiate TLS handshakes with the HAProxy.



Verify that whatever clients are participating in the TLS handshake with HAProxy have at least one cipher suite in common with HAProxy.

> **Note:** Specify cipher suites that are supported by the versions configured in the **Minimum version of TLS supported by HAProxy and Router** field.

10. Under **HAProxy forwards requests to Router over TLS**, select **Enable** or **Disable** based on your deployment layout.



- **Enable HAProxy forwarding of requests to Router over TLS**

If you want to:	Encrypt communication between HAProxy and the Gorouter
Then configure the following:	1. Leave **Enable** selected. 2. In the **Certificate Authority for HAProxy Backend** field, specify the Certificate Authority (CA) that signed the certificate you configured in the **Certificate and Private Key for HAProxy and Router** field. **Note:** If you used the **Generate RSA Certificate** link to generate a self-signed certificate, then the CA to specify is the Ops Manager CA, which you can locate at the CA endpoint in the Ops Manager API. 3. Make sure that Gorouter and HAProxy have TLS cipher suites in common in the **TLS Cipher Suites for Router** and **TLS Cipher Suites for HAProxy** fields.
See also:	◦ Terminating SSL/TLS at the Load Balancer and Gorouter ◦ Providing a Certificate for Your SSL/TLS Termination Point ◦ Using the Ops Manager API

- **Disable HAProxy forwarding of requests to Router over TLS**

If you want to:	Use non-encrypted communication between HAProxy and Gorouter, or you are not using HAProxy

Then configure the following:	1. Select **Disable**. 2. If you are not using HAProxy, set the number of HAProxy job instances to `0` on the **Resource Config** page. See [Disable Unused Resources]().
See also:	◦ [Terminating SSL/TLS at the Gorouter Only]() ◦ [Terminating SSL/TLS at the Load Balancer Only]()

11. If you are not using SSL encryption or if you are using self-signed certificates, select **Disable SSL certificate verification for this environment**. Selecting this checkbox also disables SSL verification for route services.

 Note: For production deployments, Pivotal does not recommend disabling SSL certificate verification.

12. (Optional) If you want HAProxy or the Gorouter to reject any HTTP (non-encrypted) traffic, select the **Disable HTTP on HAProxy and Gorouter** checkbox. When selected, HAProxy and Gorouter will not listen on port 80.

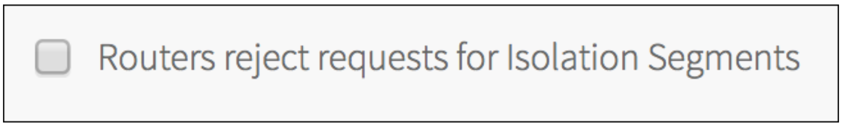
 

13. (Optional) Select the **Disable insecure cookies on the Router** checkbox to set the secure flag for cookies generated by the router.

14. (Optional) To disable the addition of Zipkin tracing headers on the Gorouter, deselect the **Enable Zipkin tracing headers on the router** checkbox. Zipkin tracing headers are enabled by default. For more information about using Zipkin trace logging headers, see [Zipkin Tracing in HTTP Headers]().

15. (Optional) To stop the Router from writing access logs to local disk, deselect the **Enable Router to write access logs locally** checkbox. You should consider disabling this checkbox for high traffic deployments since logs may not be rotated fast enough and can fill up the disk.

16. By default, the PAS routers handle traffic for applications deployed to an isolation segment created by the PCF Isolation Segment tile. To configure the PAS routers to reject requests for applications within isolation segments, select the **Routers reject requests for Isolation Segments** checkbox.

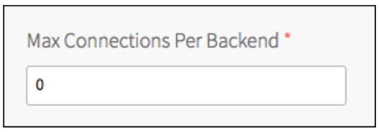
 Do not enable this option without deploying routers for each isolation segment. See the following topics for more information:
 - [Installing PCF Isolation Segment]()
 - [Sharding Routers for Isolation Segments]()

17. In the **Choose whether or not to enable route services** section, choose either **Enable route services** or **Disable route services**. Route services are a class of [marketplace services]() that perform filtering or content transformation on application requests and responses. See the [Route Services]() topic for details.

18. (Optional) If you want to limit the number of app connections to the backend, enter a value in the **Max Connections Per Backend** field. You can use this field to prevent a poorly behaving app from all the connections and impacting other apps.

 To choose a value for this field, review the peak concurrent connections received by instances of the most popular apps in your deployment. You can determine the number of concurrent connections for an app from the `httpStartStop` event metrics emitted for each app request.

 If your deployment uses PCF Metrics, you can also obtain this peak concurrent connection information from [Network Metrics](). The default value is

 `500`.

19. Enter a value for **Router Max Idle Keepalive Connections** See [Considerations for Configuring max_idle_connections]().

20. (Optional) To accommodate larger uploads over connections with high latency, increase the number of seconds in the **Router Timeout to Backends** field.

21. (Optional) Use the **Frontend Idle Timeout for Gorouter and HAProxy** field to help prevent connections from your load balancer to Gorouter or

HAProxy from being closed prematurely. The value you enter sets the duration, in seconds, that Gorouter or HAProxy maintains an idle open connection from a load balancer that supports keep-alive.

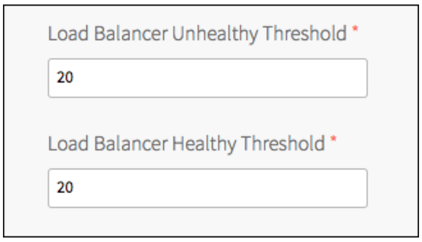
In general, set the value higher than your load balancer's backend idle timeout to avoid the race condition where the load balancer sends a request before it discovers that Gorouter or HAProxy has closed the connection.

See the following table for specific guidance and exceptions to this rule:

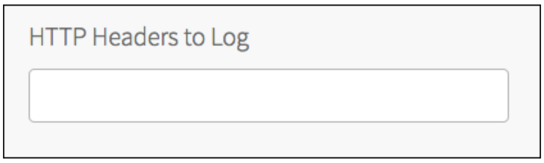
IaaS	Guidance
AWS	AWS ELB has a default timeout of 60 seconds, so Pivotal recommends a value greater than `60`.
Azure	By default, Azure load balancer times out at 240 seconds without sending a TCP RST to clients, so as an exception, Pivotal recommends a value lower than `240` to force the load balancer to send the TCP RST.
GCP	GCP has a default timeout of 600 seconds, so Pivotal recommends a value greater than `600`.
Other	Set the timeout value to be greater than that of the load balancer's backend idle timeout.

22. (Optional) Increase the value of **Load Balancer Unhealthy Threshold** to specify the amount of time, in seconds, that the router continues to accept connections before shutting down. During this period, healthchecks may report the router as unhealthy, which causes load balancers to failover to other routers. Set this value to an amount greater than or equal to the maximum time it takes your load balancer to consider a router instance unhealthy, given contiguous failed healthchecks.

23. (Optional) Modify the value of **Load Balancer Healthy Threshold**. This field specifies the amount of time, in seconds, to wait until declaring the Router instance started. This allows an external load balancer time to register the Router instance as healthy.

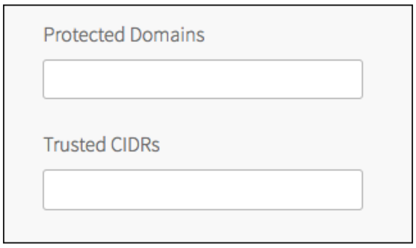


24. (Optional) If app developers in your organization want certain HTTP headers to appear in their app logs with information from the Gorouter, specify them in the **HTTP Headers to Log** field. For example, to support app developers that deploy Spring apps to PCF, you can enter Spring-specific HTTP headers.



25. If you expect requests larger than the default maximum of 16 Kbytes, enter a new value (in bytes) for **HAProxy Request Max Buffer Size**. You may need to do this, for example, to support apps that embed a large cookie or query string values in headers.

26. If your PCF deployment uses HAProxy and you want it to receive traffic only from specific sources, use the following fields:
    - **Protected Domains:** Enter a comma-separated list of domains from which PCF can receive traffic.
    - **Trusted CIDRs:** Optionally, enter a space-separated list of CIDRs to limit which IP addresses from the **Protected Domains** can send traffic to PCF.



27. For Loggregator Port, you must enter `4443`. In AWS deployments, port 4443 forwards SSL traffic that supports WebSockets from the ELB. Do not use the default port of `443`.

28. For **Container Network Plugin Interface**, ensure **Silk** is selected and review the following fields:

> **Note**: The **External** option exists to support NSX-T integration for vSphere deployments.

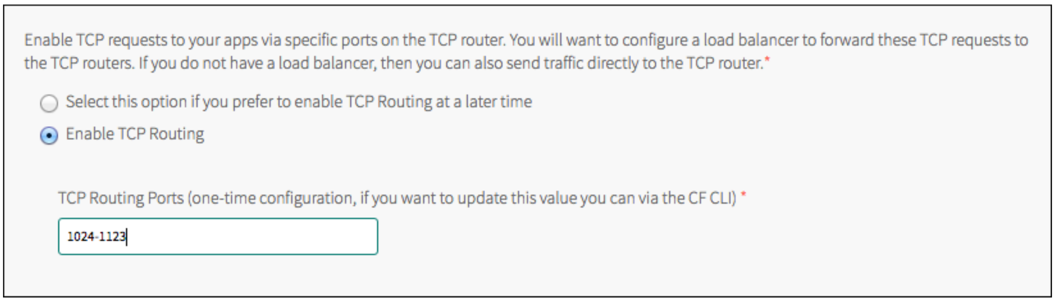
  a. (Optional) You can change the value in the **Applications Network Maximum Transmission Unit (MTU)** field. Pivotal recommends setting the MTU value for your application network to `1454`. Some configurations, such as networks that use GRE tunnels, may require a smaller MTU value.
  b. (Optional) Enter an IP range for the overlay network in the **Overlay Subnet** box. If you do not set a custom range, Ops Manager uses `10.255.0.0/16`.

  > **WARNING**: The overlay network IP range must not conflict with any other IP addresses in your network.

  c. Enter a UDP port number in the **VXLAN Tunnel Endpoint Port** box. If you do not set a custom port, Ops Manager uses 4789.
  d. For **Denied logging interval**, set the per-second rate limit for packets blocked by either a container-specific networking policy or by Application Security Group rules applied across the space, org, or deployment. This field defaults to `1`.
  e. For **UDP logging interval**, set the per-second rate limit for UDP packets sent and received. This field defaults to `100`.
  f. To enable logging for app traffic, select **Log traffic for all accepted/denied application packets**. See Manage Logging for Container-to-Container Networking for more information.

29. TCP Routing is disabled by default. To enable this feature, perform the following steps:

  a. Select **Enable TCP Routing**
  b. In **TCP Routing Ports** enter a range of ports to be allocated for TCP Routes. If you configured AWS for PCF manually, enter `1024-1123` which corresponds to the rules you created for `pcf-tcp-elb` Configuration of this field is only applied on the first deploy, and update updates to the port range are made using the cf CLI. For details about modifying the port range, see the Router Groups topic.



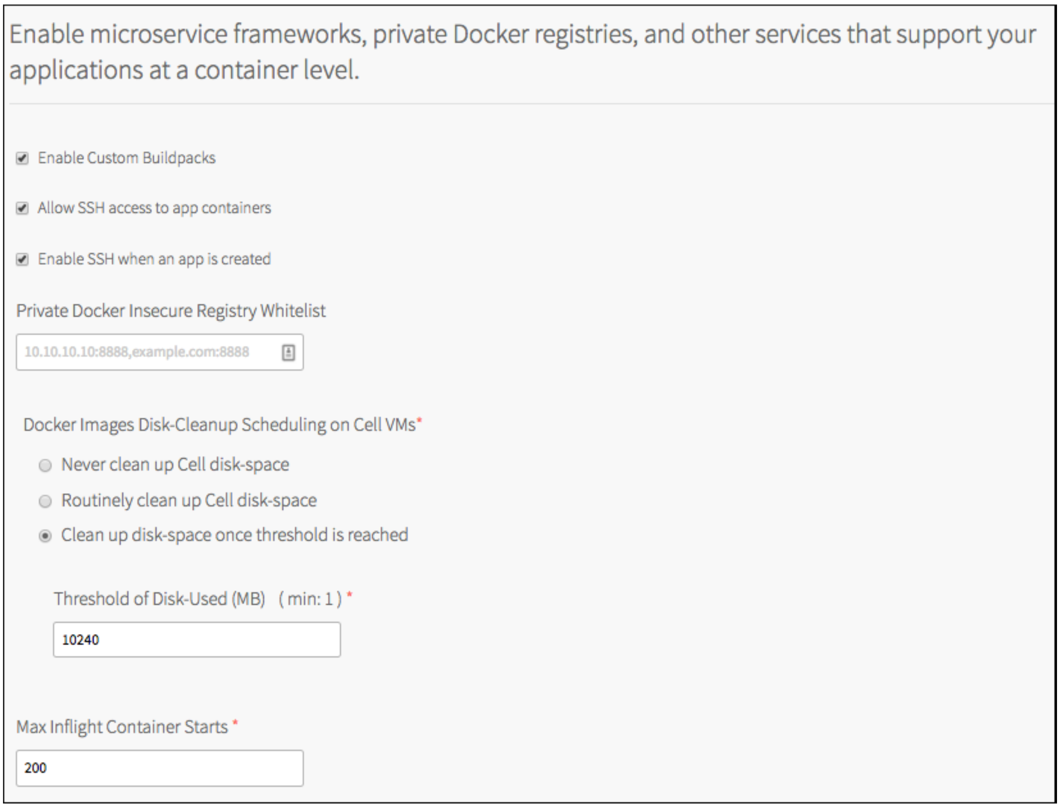
  c. For AWS, you also need to specify the name of a TCP ELB in the **LOAD BALANCER** column of TCP Router job of the `Resource Config` screen. You configure this later on in PAS. For more information, see the Configure Router to Elastic Load Balancer topic.

30. (Optional) To disable TCP routing, click **Select this option if you prefer to enable TCP Routing at a later time** For more information, see the Configuring TCP Routing in PAS topic.

31. Click **Save**

Step 5: Configure Application Containers

1. Select **Application Containers**.



2. The **Enable Custom Buildpacks** checkbox governs the ability to pass a custom buildpack URL to the `-b` option of the `cf push` command. By default, this ability is enabled, letting developers use custom buildpacks when deploying apps. Disable this option by disabling the checkbox. For more information about custom buildpacks, refer to the buildpacks section of the PCF documentation.

3. The **Allow SSH access to app containers** checkbox controls SSH access to application instances. Enable the checkbox to permit SSH access across your deployment, and disable it to prevent all SSH access. See the Application SSH Overview topic for information about SSH access permissions at the space and app scope.

4. If you want enable SSH access for new apps by default in spaces that allow SSH, select **Enable SSH when an app is created**. If you deselect the checkbox, developers can still enable SSH after pushing their apps by running `cf enable-ssh APP-NAME`.

5. You can configure Pivotal Application Service (PAS) to run app instances in Docker containers by supplying their IP address range(s) in the **Private Docker Insecure Registry Whitelist** textbox. See the Using Docker Registries topic for more information.

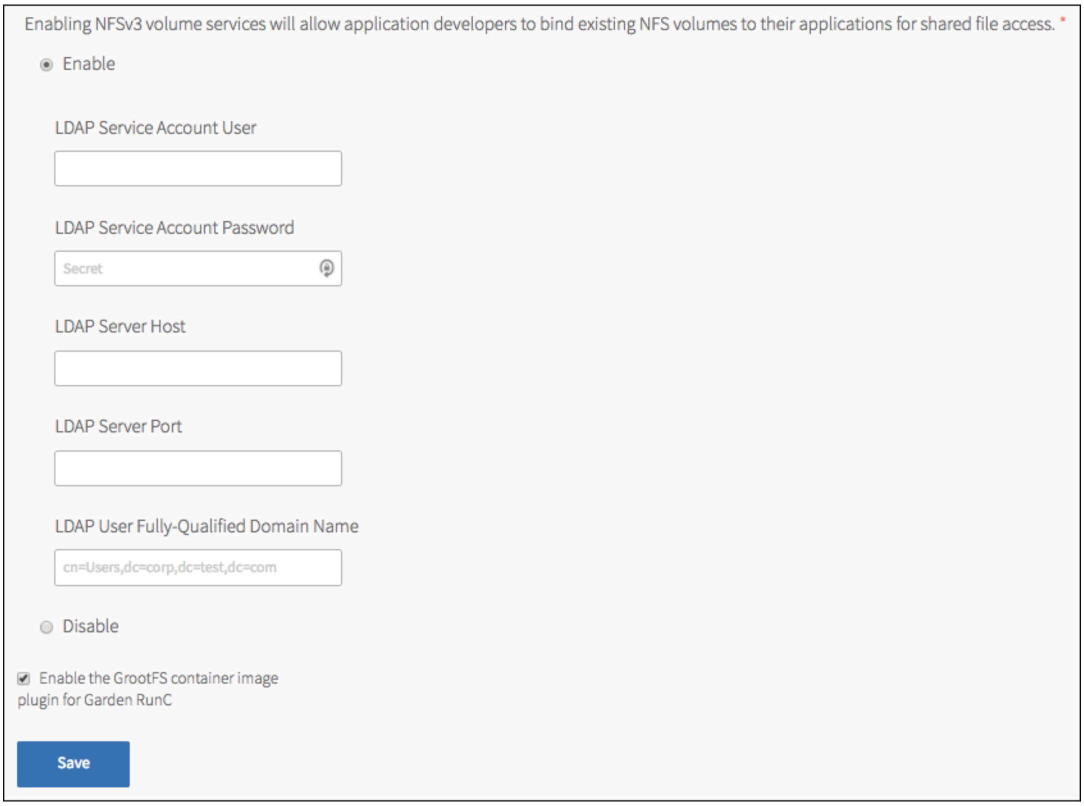
6. Select your preference for **Docker Images Disk-Cleanup Scheduling on Cell VMs**. If you choose **Clean up disk-space once threshold is reached**, enter a **Threshold of Disk-Used** in megabytes. For more information about the configuration options and how to configure a threshold, see Configuring Docker Images Disk-Cleanup Scheduling.

7. Enter a number in the **Max Inflight Container Starts** textbox. This number configures the maximum number of started instances across your deployment's Diego Cells. For more information about this feature, see Setting a Maximum Number of Started Containers.

8. Under **Enabling NFSv3 volume services**, select **Enable** or **Disable**. NFS volume services allow application developers to bind existing NFS volumes to their applications for shared file access. For more information, see the Enabling NFS Volume Services topic.

   > **Note**: In a clean install, NFSv3 volume services is enabled by default. In an upgrade, NFSv3 volume services is set to the same setting as it was in the previous deployment.

9. (Optional) To configure LDAP for NFSv3 volume services, perform the following steps:



- For **LDAP Service Account User**, enter the username of the service account in LDAP that will manage volume services.
- For **LDAP Service Account Password**, enter the password for the service account.
- For **LDAP Server Host**, enter the hostname or IP address of the LDAP server.
- For **LDAP Server Port**, enter the LDAP server port number. If you do not specify a port number, Ops Manager uses 389.
- For **LDAP Server Protocol**, enter the server protocol. If you do not specify a protocol, Ops Manager uses TCP.
- For **LDAP User Fully-Qualified Domain Name**, enter the fully qualified path to the LDAP service account. For example, if you have a service account named `volume-services` that belongs to organizational units (OU) named `service-accounts` and `my-company`, and your domain is named `domain`, the fully qualified path looks like the following:

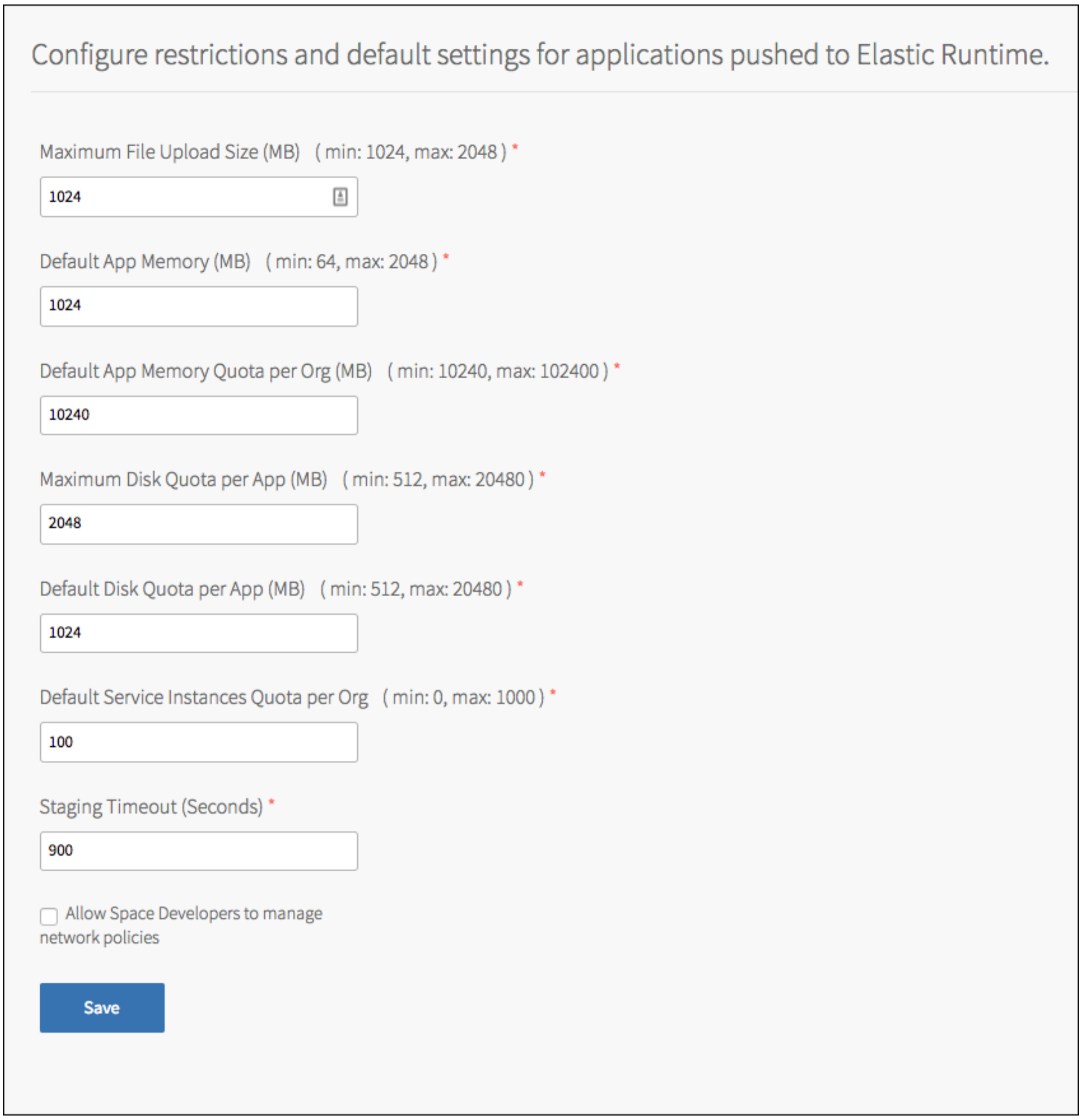
```
CN=volume-services,OU=service-accounts,OU=my-company,DC=domain,DC=com
```

10. By default, PAS manages container images using the GrootFS plugin for Garden-runC. If you experience issues with GrootFS, you can disable the plugin and use the image plugin built into Garden-runC.

11. Click **Save**.

Step 6: Configure Application Developer Controls

1. Select **Application Developer Controls**.



2. Enter the **Maximum File Upload Size (MB)**. This is the maximum size of an application upload.

3. Enter the **Default App Memory (MB)**. This is the amount of RAM allocated by default to a newly pushed application if no value is specified with the cf CLI.

4. Enter the **Default App Memory Quota per Org**. This is the default memory limit for all applications in an org. The specified limit only applies to the first installation of PAS. After the initial installation, operators can use the cf CLI to change the default value.

5. Enter the **Maximum Disk Quota per App (MB)**. This is the maximum amount of disk allowed per application.

> **Note**: If you allow developers to push large applications, PAS may have trouble placing them on Cells. Additionally, in the event of a system upgrade or an outage that causes a rolling deploy, larger applications may not successfully re-deploy if there is insufficient disk capacity. Monitor your deployment to ensure your Cells have sufficient disk to run your applications.

6. Enter the **Default Disk Quota per App (MB)**. This is the amount of disk allocated by default to a newly pushed application if no value is specified with the cf CLI.

7. Enter the **Default Service Instances Quota per Org**. The specified limit only applies to the first installation of PAS. After the initial installation, operators can use the cf CLI to change the default value .

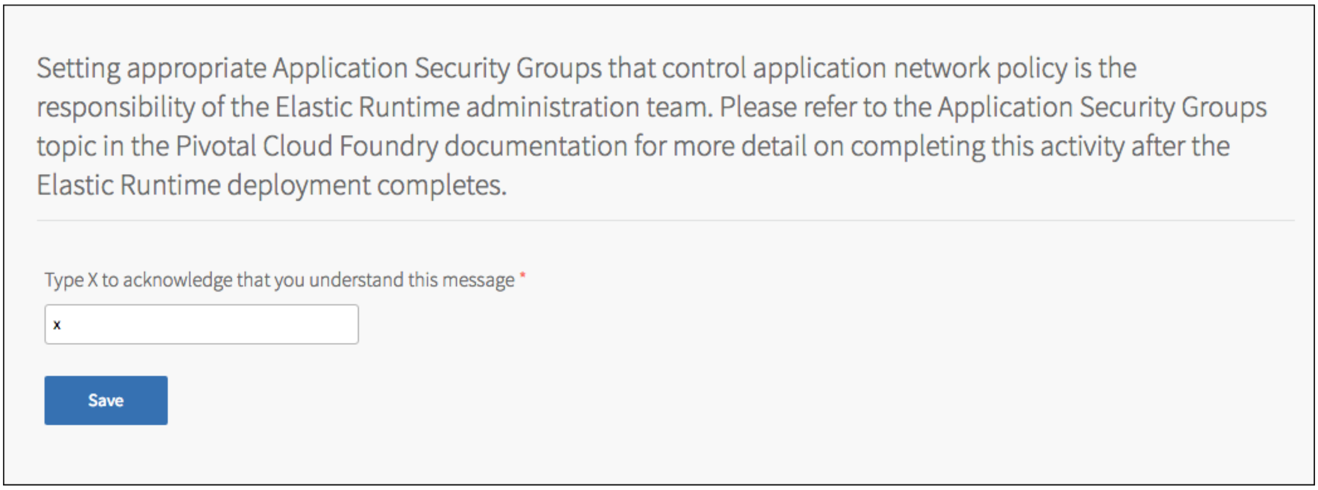
8. Enter the **Staging Timeout (Seconds)**. When you stage an application droplet with the Cloud Controller, the server times out after the number of seconds you specify in this field.

9. Select the **Allow Space Developers to manage network policies** checkbox to permit developers to manage their own network policies for their applications.

10. Click **Save**.

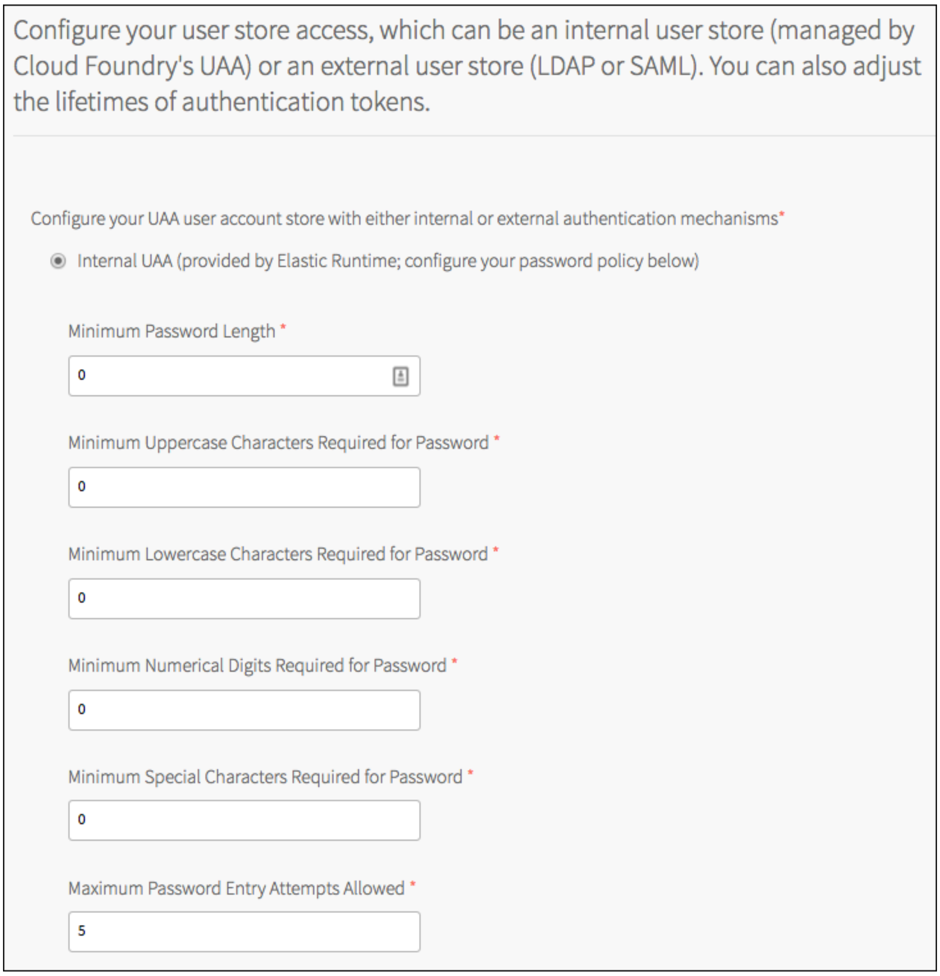
Step 7: Review Application Security Groups

Setting appropriate Application Security Groups is critical for a secure deployment. Type `X` in the box to acknowledge that once the Pivotal Application Service (PAS) deployment completes, you will review and set the appropriate application security groups. See Restricting App Access to Internal PCF Components for instructions.



Step 8: Configure Authentication and Enterprise SSO

1. Select **Authentication and Enterprise SSO**.



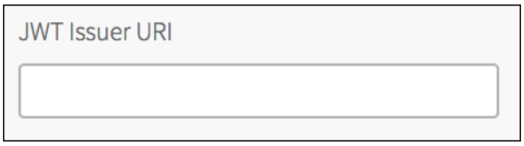
2. To authenticate user sign-ons, your deployment can use one of three types of user database: the UAA server's internal user store, an external SAML identity provider, or an external LDAP server.
   - To use the internal UAA, select the **Internal** option and follow the instructions in the Configuring UAA Password Policy topic to configure your password policy.
   - To connect to an external identity provider through SAML, scroll down to select the **SAML Identity Provider** option and follow the instructions in the Configuring PCF for SAML section of the *Configuring Authentication and Enterprise SSO for Pivotal Application Service (PAS)* topic.
   - To connect to an external LDAP server, scroll down to select the **LDAP Server** option and follow the instructions in the Configuring LDAP section of the *Configuring Authentication and Enterprise SSO for PAS* topic.

3. Click **Save**.

Step 9: Configure UAA

1. Select **UAA**.

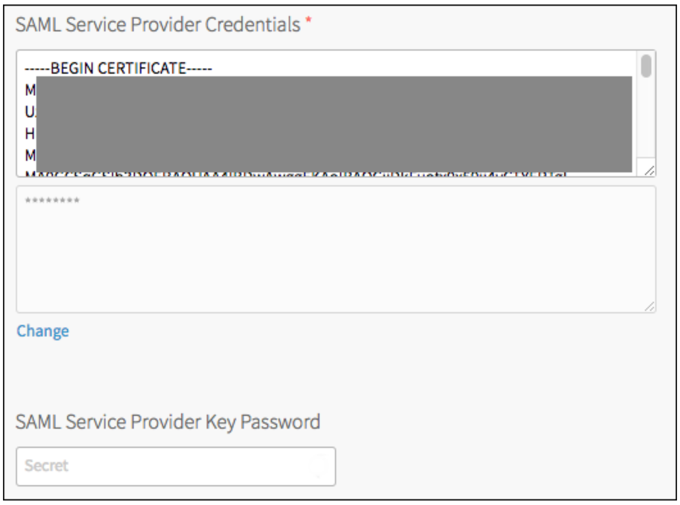
2. (Optional) Under **JWT Issuer URI**, enter the URI that UAA uses as the issuer when generating tokens.



3. Under **SAML Service Provider Credentials**, enter a certificate and private key to be used by UAA as a SAML Service Provider for signing outgoing SAML authentication requests. You can provide an existing certificate and private key from your trusted Certificate Authority or generate a self-signed certificate. The following domain must be associated with the certificate: `*.login.YOUR-SYSTEM-DOMAIN`.

   Note: The Pivotal Single Sign-On Service and Pivotal Spring Cloud Services tiles require the `*.login.YOUR-SYSTEM-DOMAIN`.

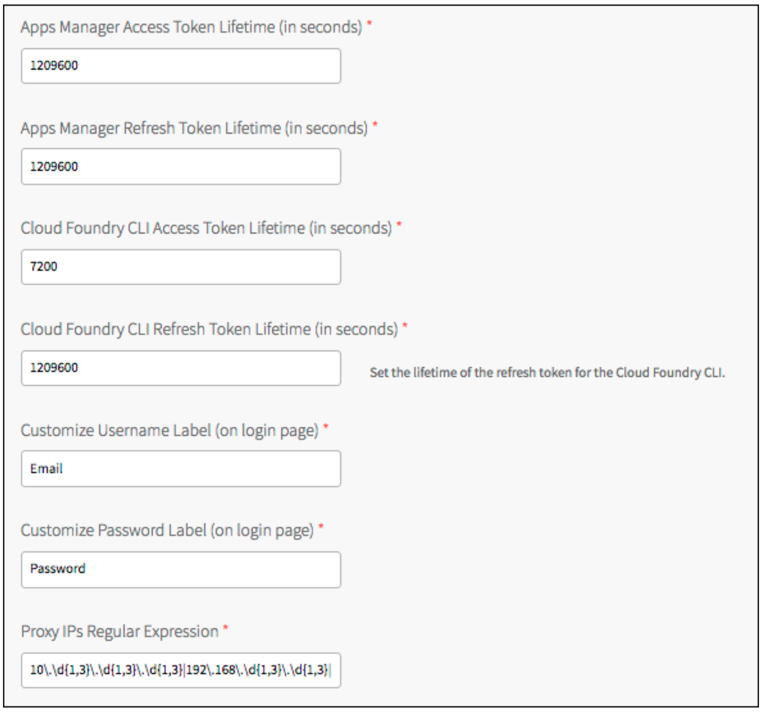
4. If the private key specified under **Service Provider Credentials** is password-protected, enter the password under **SAML Service Provider Key**



Password.

5. For **Signature Algorithm**, choose an algorithm from the dropdown menu to use for signed requests and assertions. The default value is `SHA256`.

6. (Optional) In the **Apps Manager Access Token Lifetime**, **Apps Manager Refresh Token Lifetime**, **Cloud Foundry CLI Access Token Lifetime**, and **Cloud Foundry CLI Refresh Token Lifetime** fields, change the lifetimes of tokens granted for Apps Manager and Cloud Foundry Command Line Interface (cf CLI) login access and refresh. Most deployments use the defaults.



7. (Optional) Customize the text prompts used for username and password from the cf CLI and Apps Manager login popup by entering values for **Customize Username Label (on login page)** and **Customize Password Label (on login page)**.

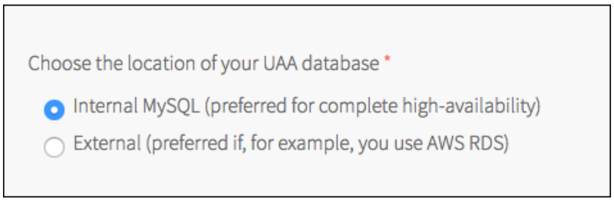
8. (Optional) The **Proxy IPs Regular Expression** field contains a pipe-delimited set of regular expressions that UAA considers to be reverse proxy IP addresses. UAA respects the `x-forwarded-for` and `x-forwarded-proto` headers coming from IP addresses that match these regular expressions. To configure UAA to respond properly to Gorouter or HAProxy requests coming from a public IP address, append a regular expression or regular expressions to match the public IP address.

9. You can configure UAA to use the internal MySQL database provided with PCF, or you can configure an external database provider. Follow the procedures in either the Internal Database Configuration or the External Database Configuration section below.

Note: If you are performing an upgrade, do not modify your existing internal database configuration or you may lose data. You must migrate your existing data before changing the configuration. See Upgrading Pivotal Cloud Foundry for additional upgrade information, and contact Pivotal Support for help.

Internal Database Configuration

1. Select **Internal MySQL**.

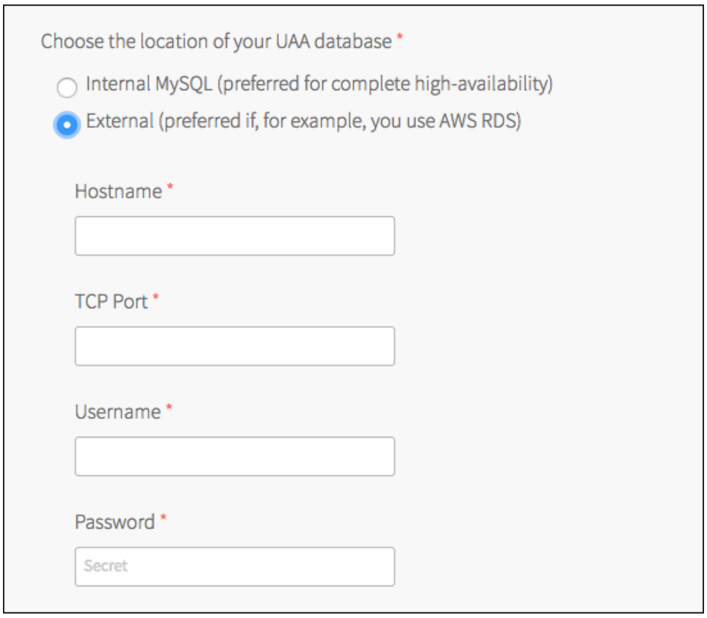


2. Click **Save**.

3. Ensure that you complete the “Configure Internal MySQL” step later in this topic to configure high availability and automatic backups for your internal MySQL databases.

External Database Configuration

1. From the **UAA** section in Pivotal Application Service (PAS), select **External**.



2. For **Hostname**, enter the hostname of the database server.

3. For **TCP Port**, enter the port of the database server.

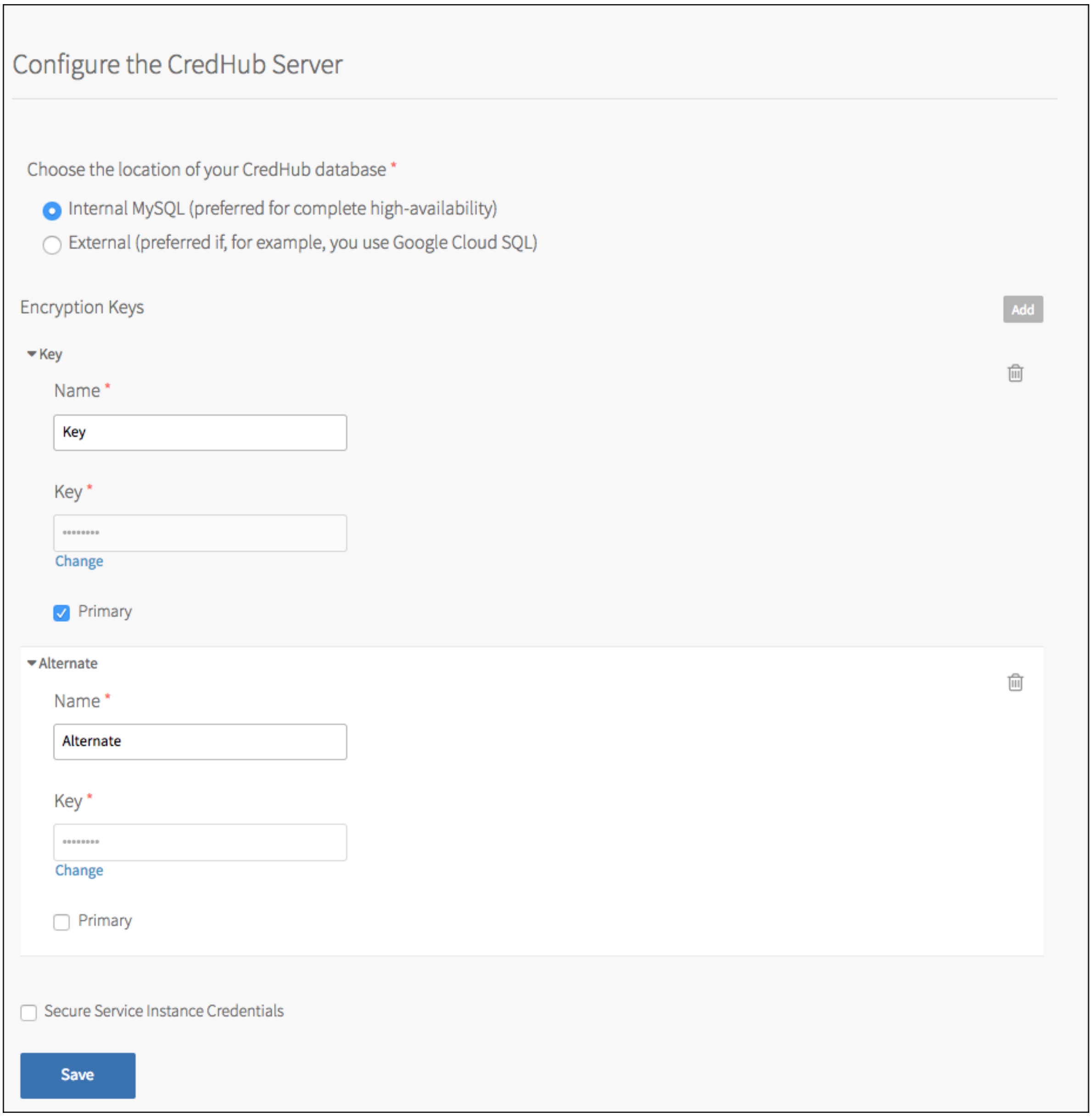
4. For **User Account and Authentication database username**, specify a unique username that can access this specific database on the database server.

5. For **User Account and Authentication database password**, specify a password for the provided username.

6. Click **Save**.

Step 10: (Optional) Configure CredHub

1. Select **Credhub**.



2. Choose the location of your CredHub Database. PAS includes this CredHub database for services to store their service instance credentials.

   a. If you chose **External**, enter the following:
      - **Hostname**: The IP address of your database server.
      - **TCP Port**: The port of your database server, such as `3306`.
      - **Username**: A unique username that can access this specific database on the database server.
      - **Password**: The password for the provided username.
      - **Database CA Certificate**: Enter a certificate to use for encrypting traffic to and from the database.

3. Under **Encryption Keys**, specify a key to use for encrypting and decrypting the values stored in the CredHub database.
   - **Name**: Enter the name of the key.
   - **Key**: Enter a key that is at least 20 characters in length.
   - **Primary**: Select this checkbox to use this key as your primary key.

   > **Note**: Ensure that you only mark one key as **Primary**. The UI includes an **Add** button to add more keys to support key rotation. For more information, see the Rotating PAS CredHub Encryption Keys topic.

4. If your deployment uses any PCF services that support storing service instance credentials in CredHub and you want to enable this feature, select the **Secure Service Instance Credentials** checkbox.

5. Select the **Resource Config** pane.

6. Under the **Job** column of the **CredHub** row, set the number of instances to `2`. This is the minimum instance count required for high availability.

Note: To use the Runtime CredHub feature, you must follow the additional steps in Securing Service Instance Credentials with Runtime CredHub.

Step 11: Configure System Databases

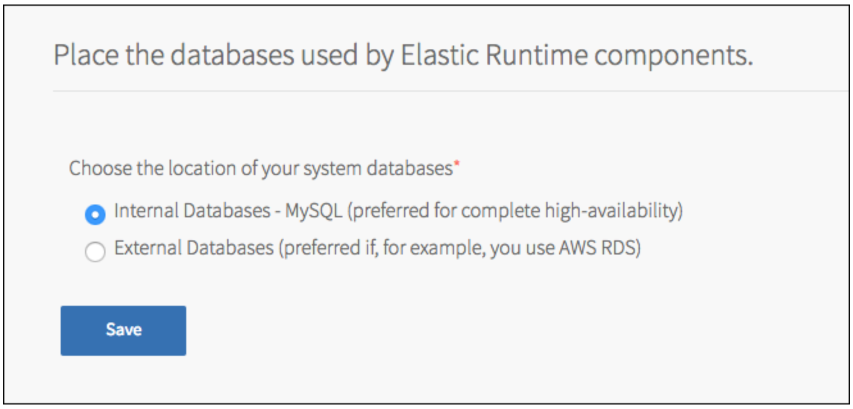
You can configure PAS to use the internal MySQL database provided with PCF, or you can configure an external database provider for the databases required by PAS.

Note: If you are performing an upgrade, do not modify your existing internal database configuration or you may lose data. You must migrate your existing data first before changing the configuration. See Upgrading Pivotal Cloud Foundry for additional upgrade information.

Internal Database Configuration

If you want to use internal databases for your deployment, perform the following steps:

1. Select **Databases**.



2. Select **Internal Databases - MySQL**.
3. Click **Save**.

Then proceed to Step 12: (Optional) Configure Internal MySQL to configure high availability and automatic backups for your internal MySQL databases.

Create External System Databases

Note: To configure an external database for UAA, see the *External Database Configuration* section of Configure UAA.

Note: The exact procedure to create databases depends upon the database provider you select for your deployment. The following procedure uses AWS RDS as an example. You can configure a different database provider that provides MySQL support, such as Google Cloud SQL.

Warning: Protect whichever database you use in your deployment with a password.

To create your Pivotal Application Service (PAS) databases, perform the following steps:

1. Add the `ubuntu` account key pair from your IaaS deployment to your local SSH profile so you can access the Ops Manager VM. For example, in AWS, you add a key pair created in AWS:

```
$ ssh-add aws-keypair.pem
```

2. SSH in to your Ops Manager using the Ops Manager FQDN and the username `ubuntu`:

```
$ ssh ubuntu@OPS-MANAGER-FQDN
```

3. Log in to your MySQL database instance using the appropriate hostname and user login values configured in your IaaS account. For example, to log

in to your AWS RDS instance, run the following MySQL command:

```
$ mysql --host=RDSHOSTNAME --user=RDSUSERNAME --password=RDSPASSWORD
```

4. Run the following MySQL commands to create databases for the eleven PAS components that require a relational database:

```
CREATE database ccdb;
CREATE database notifications;
CREATE database autoscale;
CREATE database app_usage_service;
CREATE database routing;
CREATE database diego;
CREATE database account;
CREATE database nfsvolume;
CREATE database networkpolicyserver;
CREATE database silk;
CREATE database locket;
```

5. Type `exit` to quit the MySQL client, and `exit` again to close your connection to the Ops Manager VM.

6. In PAS, select **Databases**.

7. Select the **External Databases** option.

8. For **Hostname**, enter the hostname of the database server.

9. For **TCP Port**, enter the port of the database server.

10. Each component that requires a relational database has two corresponding fields: one for the database username and one for the database password. For each set of fields, specify a unique username that can access this specific database on the database server and a password for the provided username.

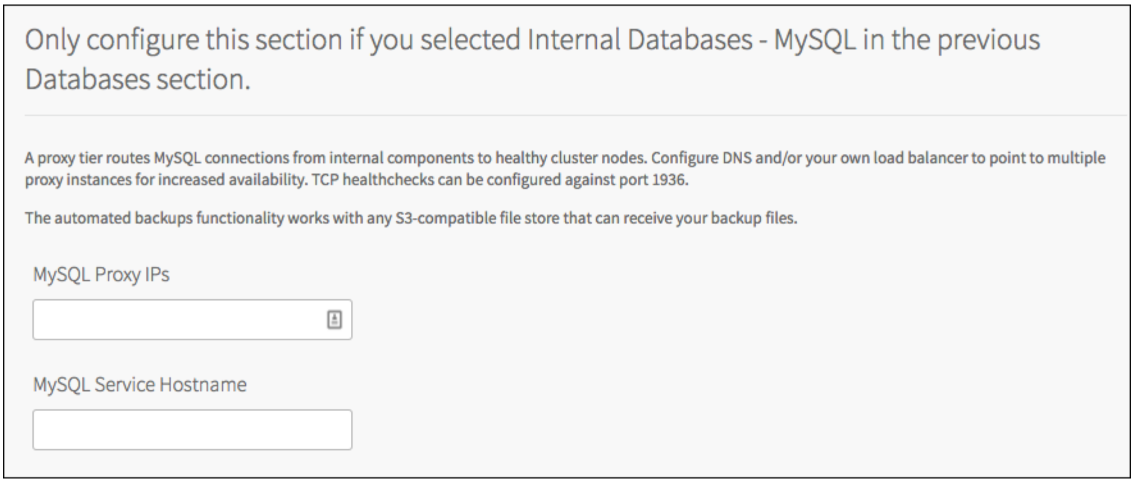
11. Click **Save**.

Step 12: (Optional) Configure Internal MySQL

Note: You only need to configure this section if you have selected **Internal Databases - MySQL** in the **Databases** section.

1. Select **Internal MySQL**.

2. In the **MySQL Proxy IPs** field, enter one or more comma-delimited IP addresses that are not in the reserved CIDR range of your network. If a MySQL node fails, these proxies re-route connections to a healthy node. See the MySQL Proxy topic for more information.



3. For **MySQL Service Hostname**, enter an IP address or hostname for your load balancer. If a MySQL proxy fails, the load balancer re-routes connections to a healthy proxy. If you leave this field blank, components are configured with the IP address of the first proxy instance entered above.

Warning: You must configure a load balancer to achieve complete high-availability.

4. In the **Replication canary time period** field, leave the default of 30 seconds or modify the value based on the needs of your deployment. Lower numbers cause the canary to run more frequently, which means that the canary reacts more quickly to replication failure but adds load to the database.

5. In the **Replication canary read delay** field, leave the default of 20 seconds or modify the value based on the needs of your deployment. This field configures how long the canary waits, in seconds, before verifying that data is replicating across each MySQL node. Clusters under heavy load can experience a small replication lag as write-sets are committed across the nodes.

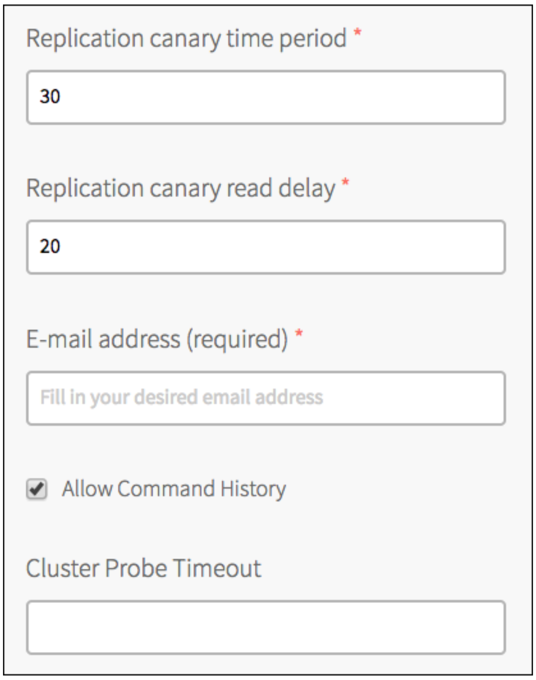
6. (**Required**): In the **E-mail address** field, enter the email address where the MySQL service sends alerts when the cluster experiences a replication issue or when a node is not allowed to auto-rejoin the cluster.

7. To prohibit the creation of command line history files on the MySQL nodes, disable the **Allow Command History** checkbox.

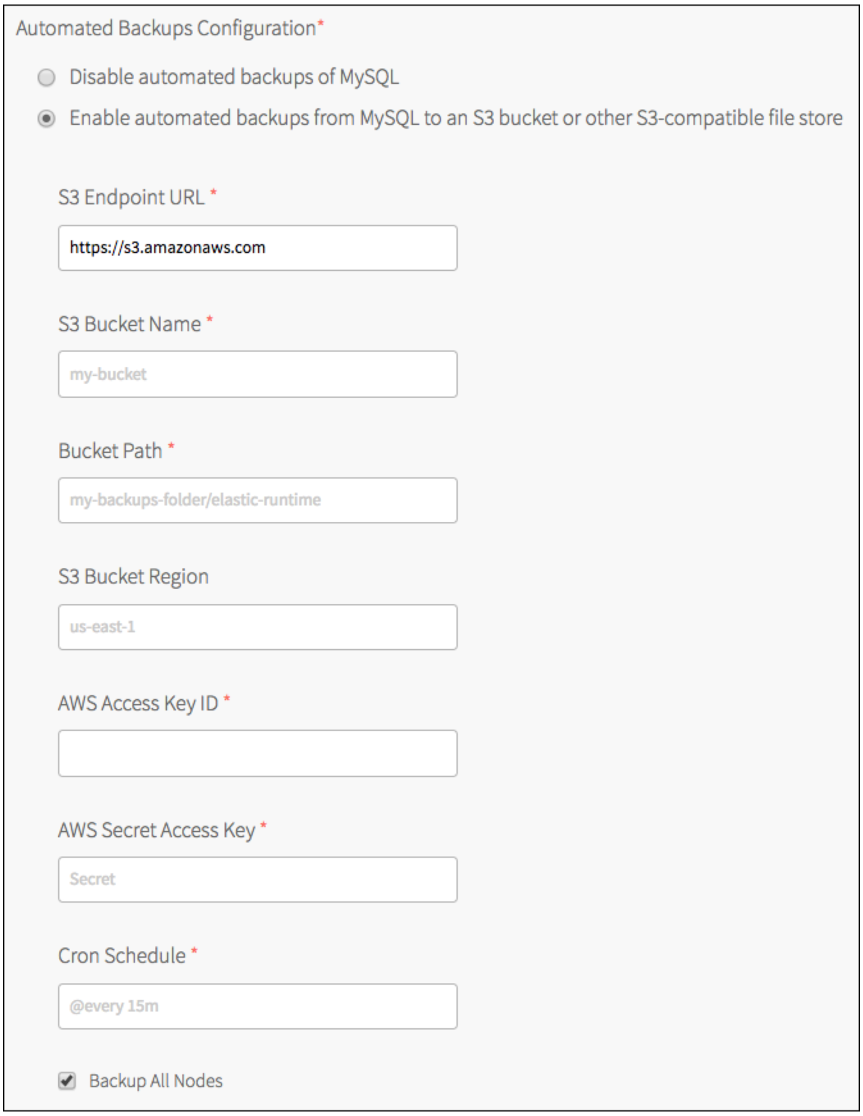
8. To allow the admin and roadmin to connect from any remote host, enable the **Allow Remote Admin Access** checkbox. When the checkbox is disabled, admins must `bosh ssh` into each MySQL VM to connect as the MySQL super user.

> **Note:** Network configuration and Application Security Groups restrictions may still limit a client's ability to establish a connection with the databases.

9. For **Cluster Probe Timeout**, enter the maximum amount of time, in seconds, that a new node will search for existing cluster nodes. If left blank, the default value is 10 seconds.



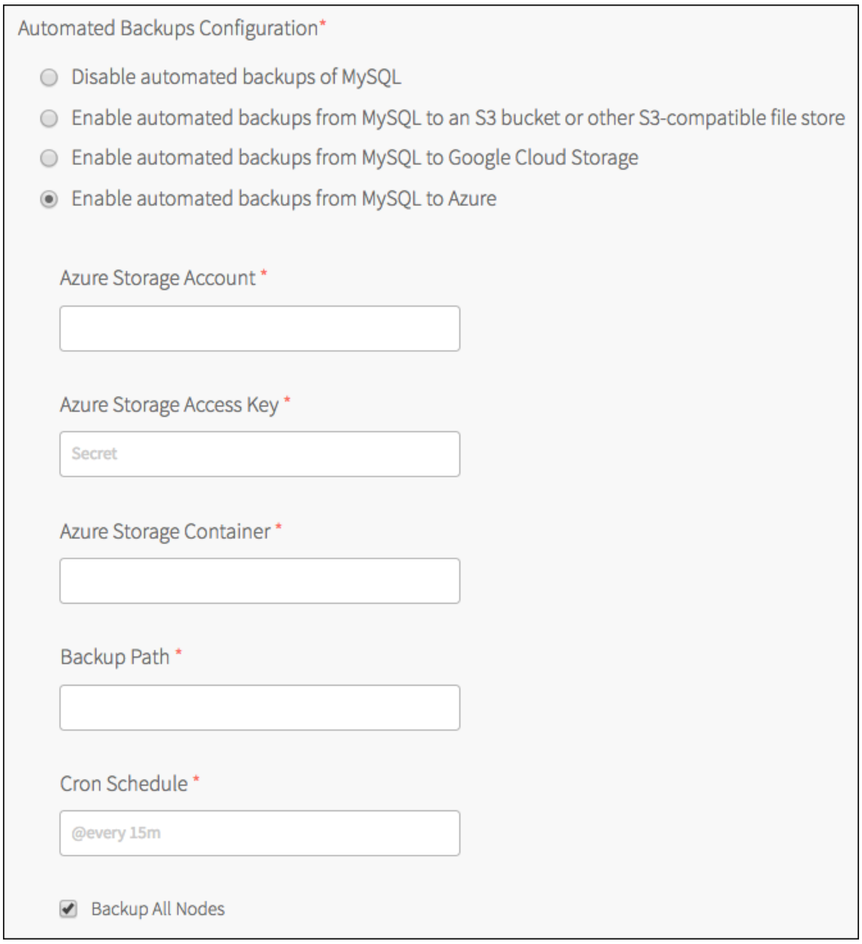
10. Under **Automated Backups Configuration**, choose one of three options for MySQL backups:
    - **Disable automatic backups of MySQL** disables automatic backups, but you can still deploy the Backup Prepare Node if you use BOSH Backup and Restore to back up your MySQL database. For more information, see the Backing Up Pivotal Cloud Foundry with BBR topic.
    - **Enable automated backups from MySQL to an S3 bucket or other S3-compatible file store** saves your backups to an existing Amazon Web Services (AWS) or Ceph S3-compatible blobstore.



This option requires the following fields:

- For **S3 Bucket Name**, enter the name of your S3 bucket. Do not include an `s3://` prefix, a trailing `/`, or underscores. If the bucket does not already exist, it will be created automatically.
- For **Bucket Path**, specify a folder within the bucket to hold your MySQL backups. Do not include a trailing `/`.
- For **S3 Bucket Region**, enter the AWS region where the bucket is located, such as `us-east-1`.
- For **AWS Access Key ID** and **AWS Secret Access Key**, enter your AWS or Ceph credentials.
- For **Cron Schedule**, enter a valid cron expression to schedule your automated backups. Cron uses your computer's local time zone.
- Enable **Backup All Nodes** to make unique backups from each instance of the MySQL server rather than just the first MySQL server instance.

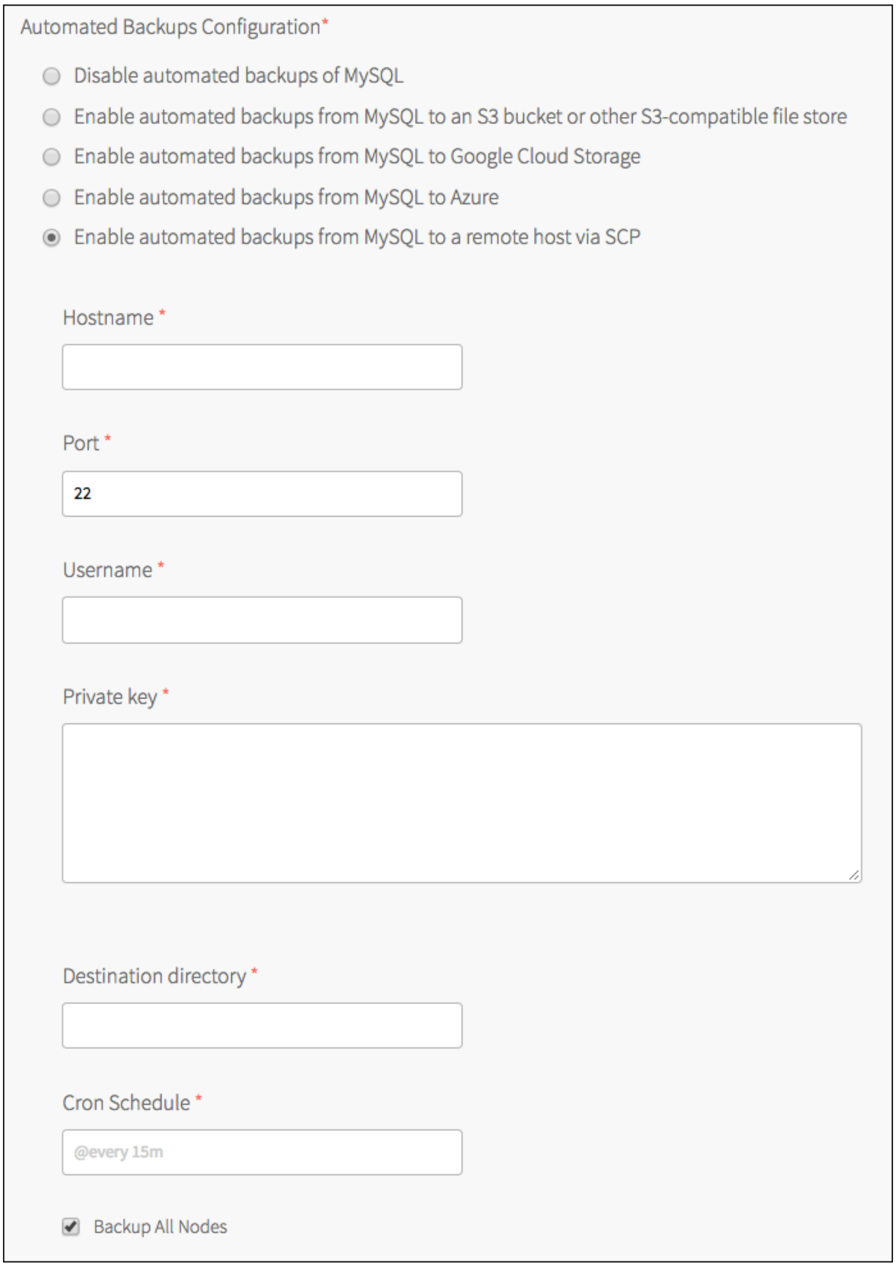
- **Enable automated backups from MySQL to Azure** saves your backups to Azure.



This option requires the following fields:

- For **Azure Storage Account**, enter the name of an existing Azure storage account where backups will be uploaded. For more information about creating and managing an Azure storage account, see the Azure documentation.
- For **Azure Storage Access Key**, enter an Azure storage access key for the storage account.
- For **Azure Storage Container**, enter the name of an existing Azure storage container that will store the backups.
- For **Backup Path**, enter the path within the Azure storage container where backups will be uploaded.
- For **Cron Schedule**, enter a valid cron expression to schedule your automated backups. Cron uses your computer's local time zone.
- Enable **Backup All Nodes** to make unique backups from each instance of the MySQL server rather than just the first MySQL server instance.

- **Enable automated backups from MySQL to a remote host via SCP**saves your backups to a remote host using secure copy protocol (SCP).



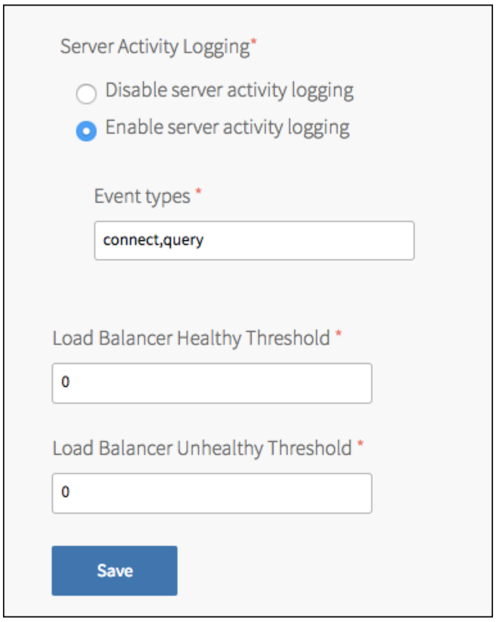
This option requires the following fields:

- For **Hostname**, enter the name of your SCP host.
- For **Port**, enter your SCP port. This should be the TCP port that your SCP host uses for SSH. The default port is `22`.
- For **Username**, enter your SSH username for the SCP host.
- For **Private key**, paste in your SSH private key.
- For **Destination directory**, enter the directory on the SCP host where you want to save backup files.
- For **Cron Schedule**, enter a valid cron expression to schedule your automated backups. Cron uses your computer's local time zone.
- Enable **Backup All Nodes** to make unique backups from each instance of the MySQL server rather than just the first MySQL server instance.

> **Note**: If you choose to enable automated MySQL backups, set the number of instances for the **Backup Prepare Node** under the **Resource Config** section of the Pivotal Application Service (PAS) tile to `1`.

11. If you want to log audit events for internal MySQL, select **Enable server activity logging** under **Server Activity Logging**.

    a. For the **Event types** field, you can enter the events you want the MySQL service to log. By default, this field includes `connect` and `query`, which tracks who connects to the system and what queries are processed. For more information, see the Logging Events section of the MariaDB documentation.



12. Enter values for the following fields:
    - **Load Balancer Healthy Threshold**: Specifies the amount of time, in seconds, to wait until declaring the MySQL proxy instance started. This allows an external load balancer time to register the instance as healthy.
    - **Load Balancer Unhealthy Threshold**: Specifies the amount of time, in seconds, that the MySQL proxy continues to accept connections before shutting down. During this period, the healthcheck reports as unhealthy to cause load balancers to fail over to other proxies. You must enter a value greater than or equal to the maximum time it takes your load balancer to consider a proxy instance unhealthy, given repeated failed healthchecks.

13. If you want to enable the MySQL interruptor feature, select the checkbox to **Prevent node auto re-join**. This feature stops all writes to the MySQL database if it notices an inconsistency in the dataset between the nodes. For more information, see the Interruptor section in the MySQL for PCF documentation.

14. Click **Save**.

Step 13: Configure File Storage

> **Note**: If you followed the instructions in Manually Configuring AWS for PCF, you created the necessary resources for external S3-compatible file storage.

To minimize system downtime, Pivotal recommends using highly resilient and redundant*external* filestores for your Pivotal Application Service (PAS) file storage.

When configuring file storage for the Cloud Controller in PAS, you can select one of the following:

- Internal WebDAV filestore
- External S3-compatible or Ceph-compatible filestore
- External Google Cloud Storage
- External Azure Cloud Storage

For production-level PCF deployments on AWS, Pivotal recommends selecting the **External S3-Compatible File Store**. For more information about production-level PCF deployments on AWS, see the Reference Architecture for Pivotal Cloud Foundry on AWS.

For more factors to consider when selecting file storage, see the Considerations for Selecting File Storage in Pivotal Cloud Foundry topic.

Internal Filestore

Internal file storage is only appropriate for small, non-production deployments.

To use the PCF internal filestore, perform the following steps:

1. In the Pivotal Application Service (PAS) tile, select **File Storage**.

2. Select **Internal WebDAV**, and click **Save**.

External S3 or Ceph Filestore

To use an external S3-compatible filestore for your Pivotal Application Service (PAS) file storage, perform the following steps:

1. In the PAS tile, select **File Storage**.

2. Select the **External S3-Compatible Filestore** option and complete the following fields:
   - Prepend `https://` to the endpoint for your region and copy it into the Ops Manager **URL Endpoint** field. For example, in the **us-west-2** region, use `https://s3-us-west-2.amazonaws.com/`.
   - Enter the **Access Key** and **Secret Key** of the `pcf-user` you created when configuring AWS for PCF.
   - For **S3 Signature Version** and **Region**, use the **V4 Signature** values. AWS recommends using Signature Version 4.
   - For **Region**, enter the region in which your S3 buckets are located, such as `us-west-2`.
   - Select **Server-side Encryption (available for AWS S3 only)** to encrypt the contents of your S3 filestore.
   - Enter values for the remaining fields as follows:

Ops Manager Field	Value	Description
Buildpacks Bucket Name	`pcf-buildpacks-bucket`	This S3 bucket stores app buildpacks.
Droplets Bucket Name	`pcf-droplets-bucket`	This S3 bucket stores app droplets. Pivotal recommends that you use a unique bucket name for droplets, but you can also use the same name as above.
Packages Bucket Name	`pcf-packages-bucket`	This S3 bucket stores app packages. Pivotal recommends that you use a unique bucket name for packages, but you can also use the same name as above.
Resources Bucket Name	`pcf-resources-bucket`	This S3 bucket stores app resources. Pivotal recommends that you use a unique bucket name for app resources, but you can also use the same name as above.

3. Click **Save**.

Note: For more information regarding AWS S3 Signatures, see the Authenticating Requests topic in the AWS documentation.

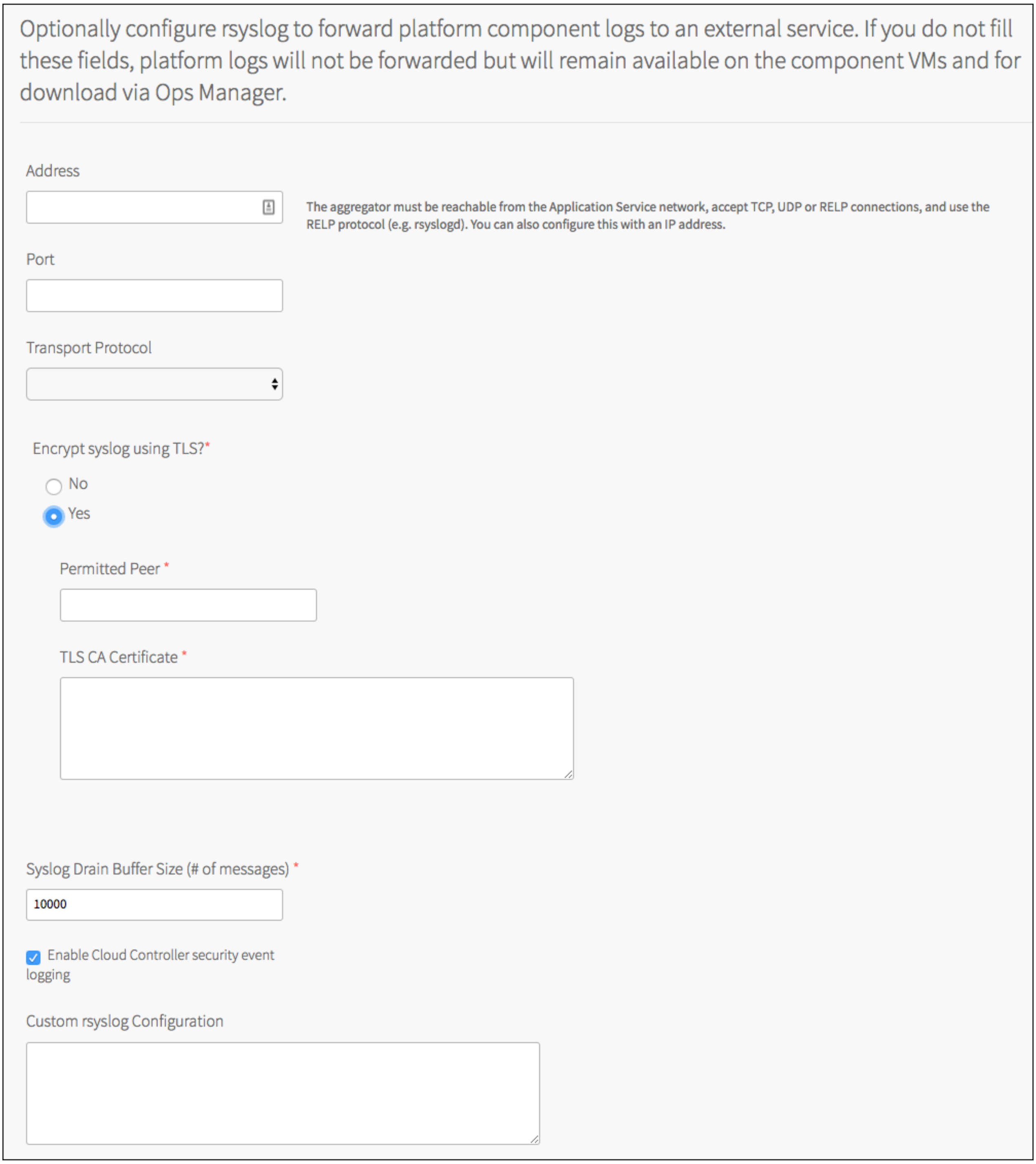
Other IaaS Storage Options

Google Cloud Storage and Azure Storage are also available as file storage options, but are not recommended for typical PCF on AWS installations.

Step 14: (Optional) Configure System Logging

If you forward logging messages to an external Reliable Event Logging Protocol (RELP) server, complete the following steps:

1. Select the **System Logging** section that is located within your PAS **Settings** tab.



2. Enter the IP address of your syslog server in **Address**.

3. Enter the port of your syslog server in **Port**. The default port for a syslog server is `514`.

> **Note**: The host must be reachable from the PAS network, accept TCP connections, and use the RELP protocol. Ensure your syslog server listens on external interfaces.

4. Select a **Transport Protocol** to use when forwarding logs.

5. If you plan to use TLS encryption when sending logs to the remote server, select **Yes** when answering the **Encrypt syslog using TLS?** question.

    a. In the **Permitted Peer** field, enter either the name or SHA1 fingerprint of the remote peer.
    b. In the **TLS CA Certificate** field, enter the TLS CA Certificate for the remote server.

6. For the **Syslog Drain Buffer Size**, enter the number of messages the Doppler server can hold from Metron agents before the server starts to drop them. See the Loggregator Guide for Cloud Foundry Operators topic for more details.

7. If you want to include security events in your log stream, select the **Enable Cloud Controller security event logging** checkbox. This logs all API requests, including the endpoint, user, source IP address, and request result, in the Common Event Format (CEF).

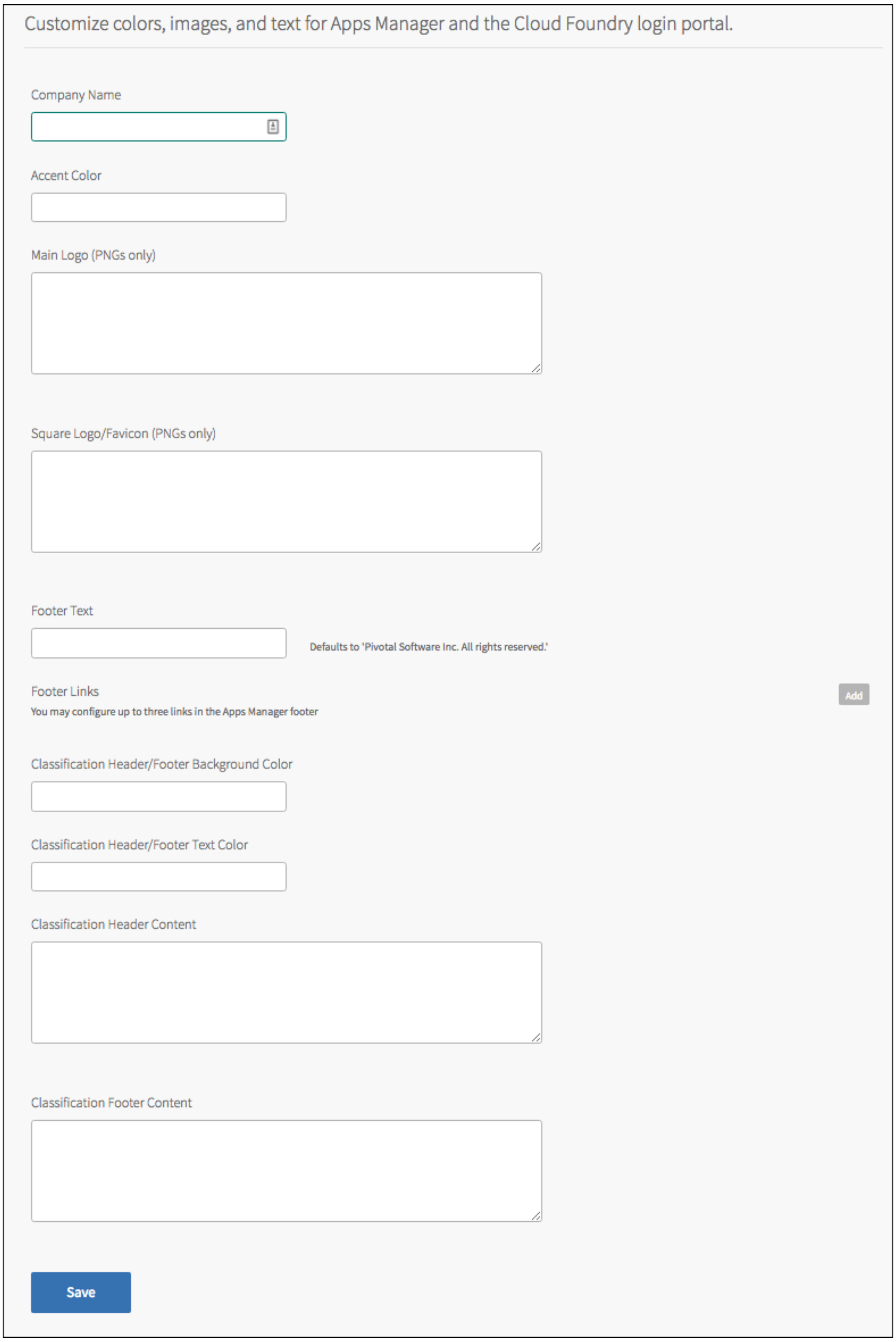
8. If you want to specify a custom syslog formatting rule, enter it in the **Custom rsyslog Configuration** field in RainerScript syntax.

9. Click **Save**.

Step 15: (Optional) Customize Apps

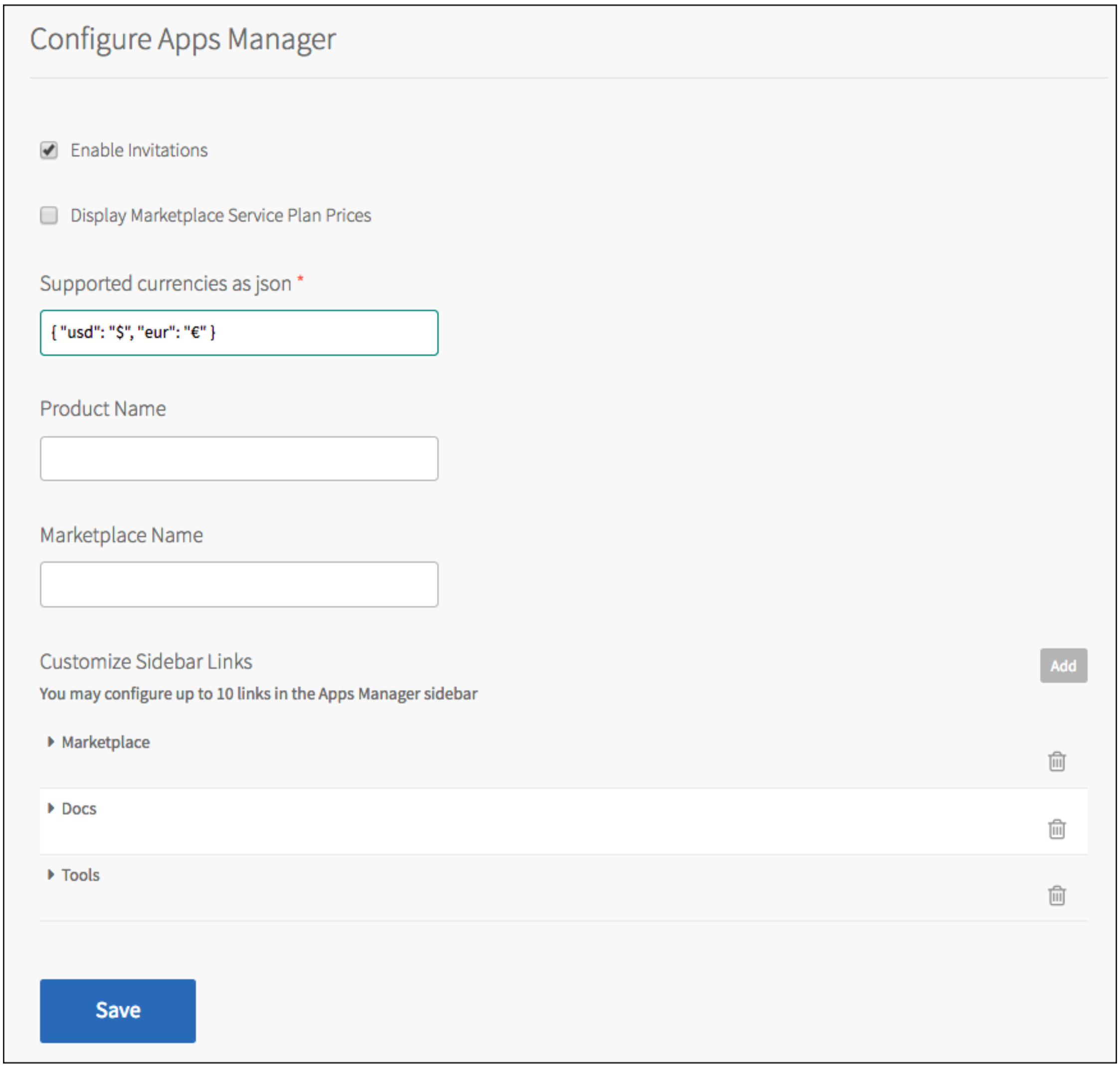
The **Custom Branding** and **Apps Manager** sections customize the appearance and functionality of Apps Manager. Refer to Custom Branding Apps Manager for descriptions of the fields on these pages and for more information about customizing Apps Manager.

1. Select **Custom Branding**. Use this section to configure the text, colors, and images of the interface that developers see when they log in, create an account, reset their password, or use Apps Manager.



2. Click **Save** to save your settings in this section.

3. Select **Apps Manager**.



4. Select **Enable Invitations** to enable invitations in Apps Manager. Space Managers can invite new users for a given space, Org Managers can invite new users for a given org, and Admins can invite new users across all orgs and spaces. See the Inviting New Users section of the *Managing User Roles with Apps Manager* topic for more information.

5. Select **Display Marketplace Service Plan Prices** to display the prices for your services plans in the Marketplace.

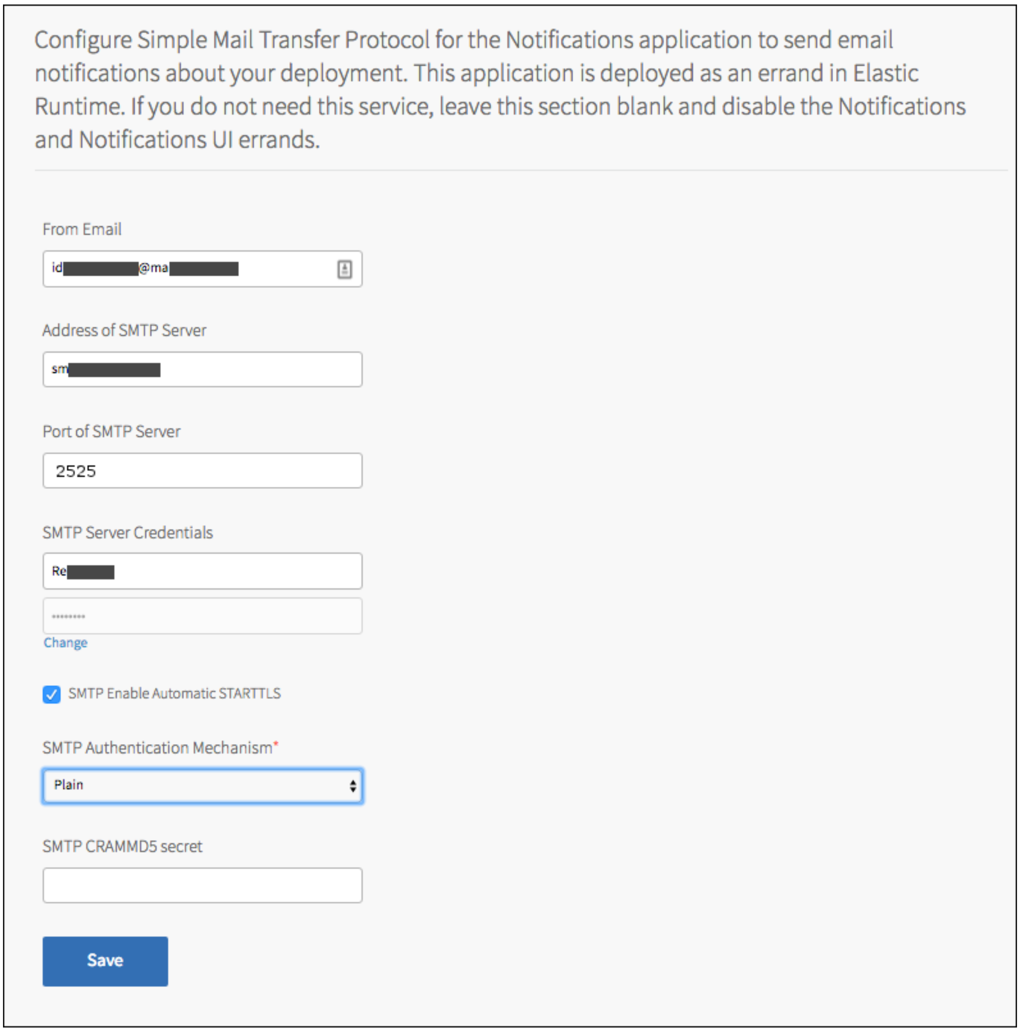
6. Enter the **Supported currencies as json** to appear in the Marketplace. Use the format `{"CURRENCY-CODE":"SYMBOL"}`. This defaults to `{"usd": "$", "eur": "€"}`.

7. Use **Product Name**, **Marketplace Name**, and **Customize Sidebar Links** to configure page names and sidebar links in the **Apps Manager** and **Marketplace** pages.

8. Click **Save** to save your settings in this section.

Step 16: (Optional) Configure Email Notifications

PAS uses SMTP to send invitations and confirmations to Apps Manager users. You must complete the **Email Notifications** page if you want to enable end-user self-registration.

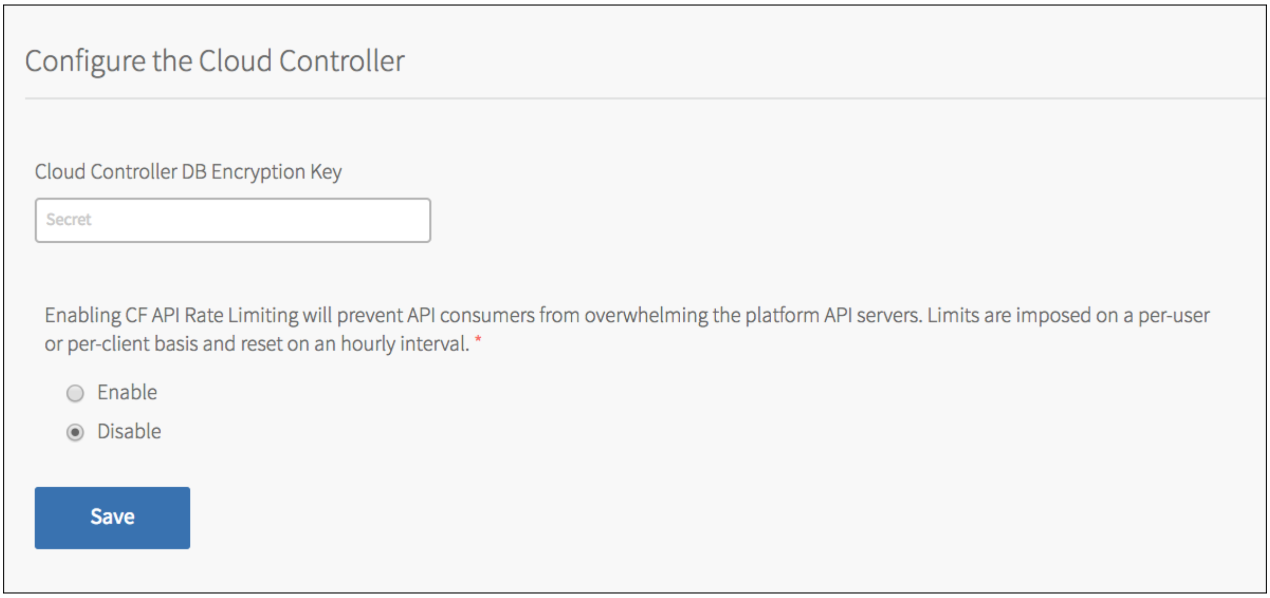


1. Select **Email Notifications**.
2. Enter your reply-to and SMTP email information
3. For **SMTP Authentication Mechanism**, select `None`.
4. Click **Save**.

Note: If you do not configure the SMTP settings using this form, the administrator must create orgs and users using the cf CLI tool. See Creating and Managing Users with the cf CLI for more information.

Step 17: Configure Cloud Controller

1. Click **Cloud Controller**.



2. Enter your **Cloud Controller DB Encryption Key** if all of the following are true:
   - You deployed Pivotal Application Service (PAS) previously.
   - You then stopped PAS or it crashed.
   - You are re-deploying PAS with a backup of your Cloud Controller database.

   See Backing Up Pivotal Cloud Foundry for more information.

3. CF API Rate Limiting prevents API consumers from overwhelming the platform API servers. Limits are imposed on a per-user or per-client basis and reset on an hourly interval.

   To disable CF API Rate Limiting, select **Disable** under **Enable CF API Rate Limiting**. To enable CF API Rate Limiting, perform the following steps:

   a. Under **Enable CF API Rate Limiting**, select **Enable**.
   b. For **General Limit**, enter the number of requests a user or client is allowed to make over an hour interval for all endpoints that do not have a custom limit. The default value is `2000`.
   c. For **Unauthenticated Limit**, enter the number of requests an unauthenticated client is allowed to make over an hour interval. The default value is `100`.

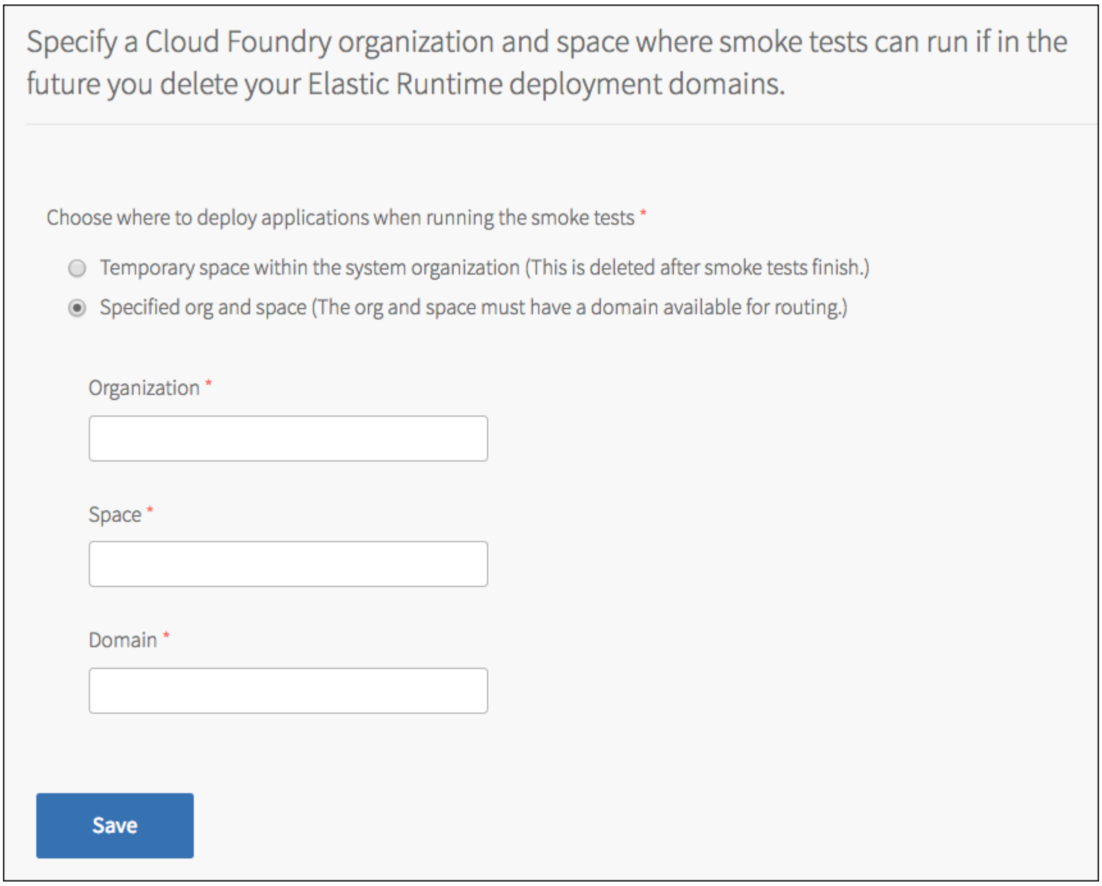
4. Click **Save**.

Step 18: Configure Smoke Tests

The Smoke Tests errand runs basic functionality tests against your Pivotal Application Service (PAS) deployment after an installation or update. In this section, choose where to run smoke tests. In the **Errands** section, you can choose whether or not to run the Smoke Tests errand.

1. Select **Smoke Tests**.

2. If you have a shared apps domain, select **Temporary space within the system organization**, which creates a temporary space within the `system` organization for running smoke tests and deletes the space afterwards. Otherwise, select **Specified org and space** and complete the fields to specify where you want to run smoke tests.



3. Click **Save**.

Step 19: (Optional) Enable Advanced Features

The **Advanced Features** section of Pivotal Application Service (PAS) includes new functionality that may have certain constraints. Although these features are fully supported, Pivotal recommends caution when using them in production environments.

Diego Cell Memory and Disk Overcommit

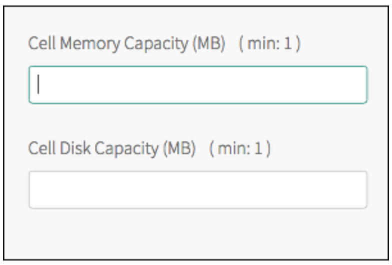
If your apps do not use the full allocation of disk space and memory set in the **Resource Config** tab, you might want use this feature. These fields control the amount to overcommit disk and memory resources to each Diego Cell VM.

For example, you might want to use the overcommit if your apps use a small amount of disk and memory capacity compared to the amounts set in the **Resource Config** settings for **Diego Cell**.

> **Note**: Due to the risk of app failure and the deployment-specific nature of disk and memory use, Pivotal has no recommendation about how much, if any, memory or disk space to overcommit.

To enable overcommit, follow these steps:

1. Select **Advanced Features**.



2. Enter the total desired amount of Diego cell memory value in the **Cell Memory Capacity (MB)** field. Refer to the **Diego Cell** row in the **Resource Config** tab for the current Cell memory capacity settings that this field overrides.

3. Enter the total desired amount of Diego cell disk capacity value in the **Cell Disk Capacity (MB)** field. Refer to the **Diego Cell** row in the **Resource Config** tab for the current Cell disk capacity settings that this field overrides.

4. Click **Save**.

> **Note**: Entries made to each of these two fields set the total amount of resources allocated, not the overage.

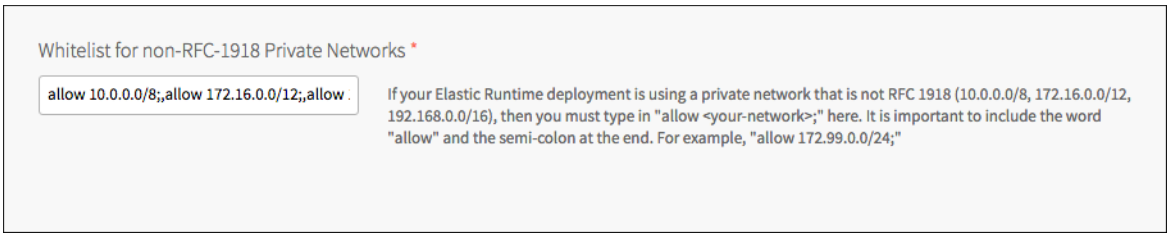
Whitelist for Non-RFC-1918 Private Networks

Some private networks require extra configuration so that internal file storage (WebDAV) can communicate with other PCF processes.

The **Whitelist for non-RFC-1918 Private Networks** field is provided for deployments that use a non-RFC 1918 private network. This is typically a private network other than `10.0.0.0/8`, `172.16.0.0/12`, or `192.168.0.0/16`.

Most PCF deployments do not require any modifications to this field.

To add your private network to the whitelist, perform the following steps:

1. Select **Advanced Features**.
2. Append a new `allow` rule to the existing contents of the **Whitelist for non-RFC-1918 Private Networks** field.


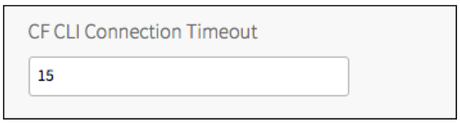
Include the word `allow`, the network CIDR range to allow, and a semi-colon (`;`) at the end. For example: `allow 172.99.0.0/24;`
3. Click **Save**.

CF CLI Connection Timeout

The **CF CLI Connection Timeout** field allows you to override the default five second timeout of the Cloud Foundry Command Line Interface (cf CLI) used within your PCF deployment. This timeout affects the cf CLI command used to push PAS errand apps such as Notifications, Autoscaler, and Apps Manager.

Set the value of this field to a higher value, in seconds, if you are experiencing domain name resolution timeouts when pushing errands in PAS.

To modify the value of the **CF CLI Connection Timeout**, perform the following steps:

1. Select **Advanced Features**.
2. Add a value, in seconds, to the **CF CLI Connection Timeout** field.


3. Click **Save**.

Step 20: Configure Errands

Errands are scripts that Ops Manager runs automatically when it installs or uninstalls a product, such as a new version of Pivotal Application Service (PAS). There are two types of errands: *post-deploy errands* run after the product is installed, and *pre-delete errands* run before the product in uninstalled.

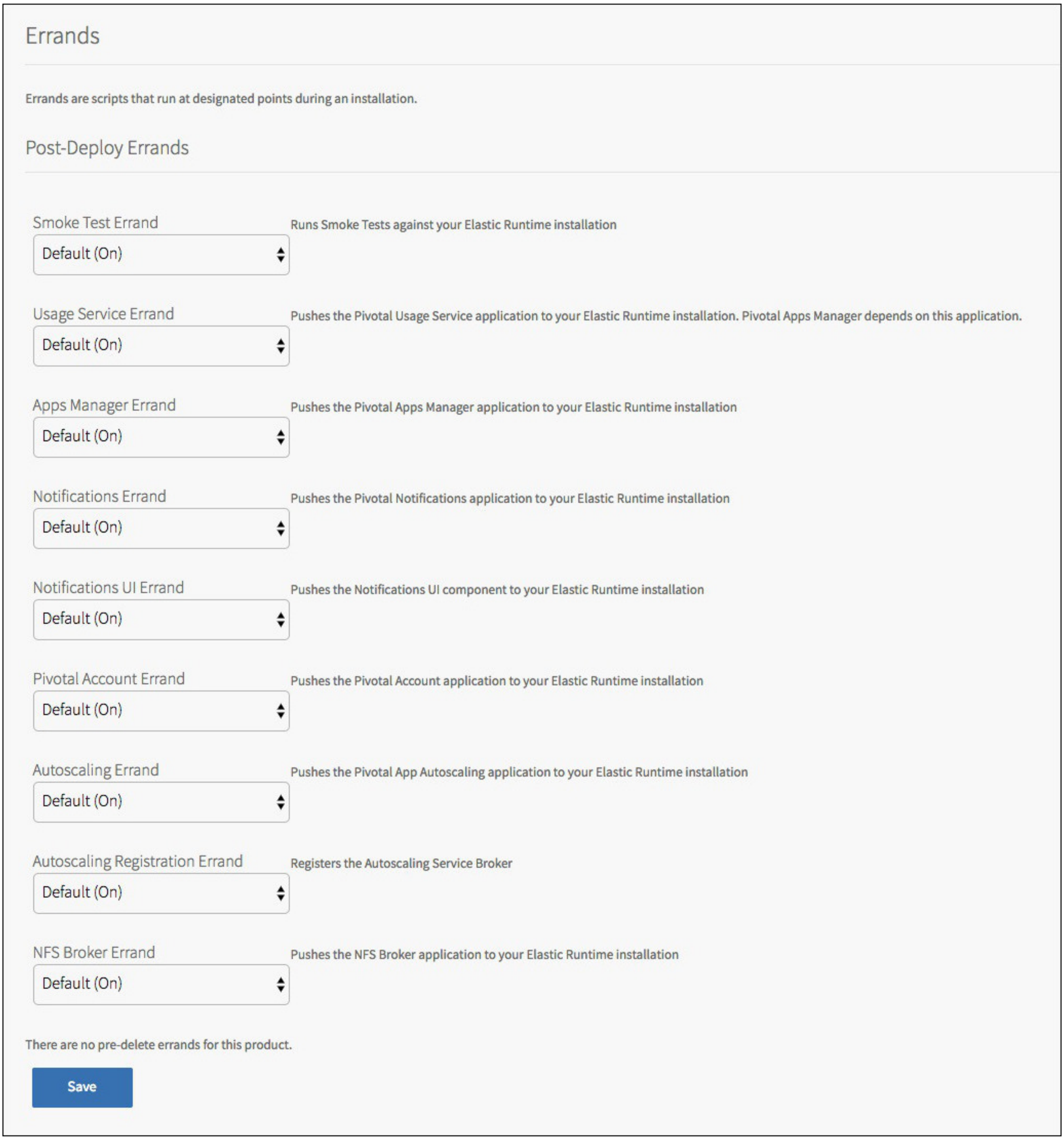
By default, Ops Manager always runs pre-delete errands, and only runs post-deploy errands when the product has changed since the last time Ops

Manager installed something. In PAS, the Smoke Test Errand defaults to always run.

The PAS tile **Errands** pane lets you change these run rules. For each errand, you can select **On** to run it always, **Off** to never run it, or **When Changed** to run it only when the product has changed since the last install.

For more information about how Ops Manager manages errands, see the Managing Errands in Ops Manager topic.

Note: Several errands deploy apps that provide services for your deployment, such as Autoscaling and Notifications. Once one of these apps is running, selecting **Off** for the corresponding errand on a subsequent installation does not stop the app.



- **Smoke Test Errand** verifies that your deployment can do the following:
  - Push, scale, and delete apps
  - Create and delete orgs and spaces
- **Usage Service Errand** deploys the Pivotal Usage Service application, which Apps Manager depends on.
- **Apps Manager Errand** deploys Apps Manager, a dashboard for managing apps, services, orgs, users, and spaces. Until you deploy Apps Manager, you

must perform these functions through the cf CLI. After Apps Manager has been deployed, Pivotal recommends deselecting the checkbox for this errand on subsequent PAS deployments. For more information about Apps Manager, see the Getting Started with the Apps Manager topic.

- **Notifications Errand** deploys an API for sending email notifications to your PCF platform users.

> **Note**: The Notifications app requires that you configure SMTP with a username and password, even if you set the value of **SMTP Authentication Mechanism** to `none`.

- **Notifications UI Errand** deploys a dashboard for users to manage notification subscriptions.
- **Pivotal Account Errand** deploys Pivotal Account, a dashboard that allows users to create and manage their accounts. In the Pivotal Account dashboard, users can launch applications, manage their profiles, manage account security, manage notifications, and manage approvals. See the Enabling Pivotal Account topic for more information.
- **Autoscaling Errand** enables you to configure your apps to automatically scale in response to changes in their usage load. See the Scaling an Application Using Autoscaler topic for more information.
- **Autoscaling Registration Errand** makes the Autoscaling service available to your applications. Without this errand, you cannot bind the Autoscaling app to your apps.
- **NFS Broker Errand** enables you to use NFS Volume Services by installing the NFS Broker app in PAS. See the Enabling NFS Volume Services topic for more information.

Step 21: (Optional) Disable Unused Resources

> **Note**: The **Resource Config** pane has fewer VMs if you are installing the Small Footprint Runtime.

> **Note**: The Small Footprint Runtime does not default to a highly available configuration. It defaults to the minimum configuration. If you want to make the Small Footprint Runtime highly available, scale the **Compute**, **Router**, and **Database** VMs to `3` instances and scale the **Control** VM to `2` instances.

Pivotal Application Service (PAS) defaults to a highly available resource configuration. However, you may need to perform additional procedures to make your deployment highly available. See the Zero Downtime Deployment and Scaling in CF and the Scaling Instances in PAS topics for more information.

If you do not want a highly available resource configuration, you must scale down your instances manually by navigating to the **Resource Config** section and using the drop-down menus under **Instances** for each job.

By default, PAS also uses an internal filestore and internal databases. If you configure PAS to use external resources, you can disable the corresponding system-provided resources in Ops Manager to reduce costs and administrative overhead.

Complete the following procedures to disable specific VMs in Ops Manager:

1. Click **Resource Config**.
2. If you configure PAS to use an external S3-compatible filestore, edit the following fields:
   - **File Storage**: Enter `0` in **Instances**.
3. If you selected **External** when configuring the UAA or System databases, edit the following fields:
   - **MySQL Proxy**: Enter `0` in **Instances**.
   - **MySQL Server**: Enter `0` in **Instances**.
   - **MySQL Monitor**: Enter `0` in **Instances**.
   - **Cloud Controller Database**: Enter `0` in **Instances**.
   - **UAA Database**: Enter `0` in **Instances**.
4. If you are not using HAProxy, enter `0` in the **Instances** field for **HAProxy**.
5. Click **Save**.

Step 22: Configure Router to Elastic Load Balancers

1. Record the names of your ELBs. If you followed the procedures in the Manually Configuring AWS for PCF topic, you created the following:
   - `pcf-ssh-elb`: A SSH load balancer
   - `pcf-web-elb`: A web load balancer
   - `pcf-tcp-elb`: A TCP load balancer

2. In the PAS tile, click **Resource Config**.

3. Enter the name of you SSH load balancer depending on which release you are using. You can specify multiple load balancers by entering the names separated by commas.
   - **PAS**: In the **ELB Name** field of the **Diego Brain** row, enter the name of your SSH load balancer: `pcf-ssh-elb`.
   - **Small Footprint Runtime**: In the **ELB Name** field of the **Control** row, enter the name of your SSH load balancer: `pcf-ssh-elb`.

4. In the **ELB Name** field of the **Router** row, enter the name of your web load balancer: `pcf-web-elb`. You can specify multiple load balancers by entering the names separated by commas.

   > **Note**: If you are using HAProxy in your deployment, then put the name of the load balancers in the **ELB Name** field of the **HAProxy** row instead of the **Router** row. For a high availability configuration, scale up the HAProxy job to more than one instance.

5. In the **ELB Name** field of the **TCP Router** row, enter the name of your TCP load balancer if you enabled TCP routing: `pcf-tcp-elb`. You can specify multiple load balancers by entering the names separated by commas.

Step 23: Download Stemcell

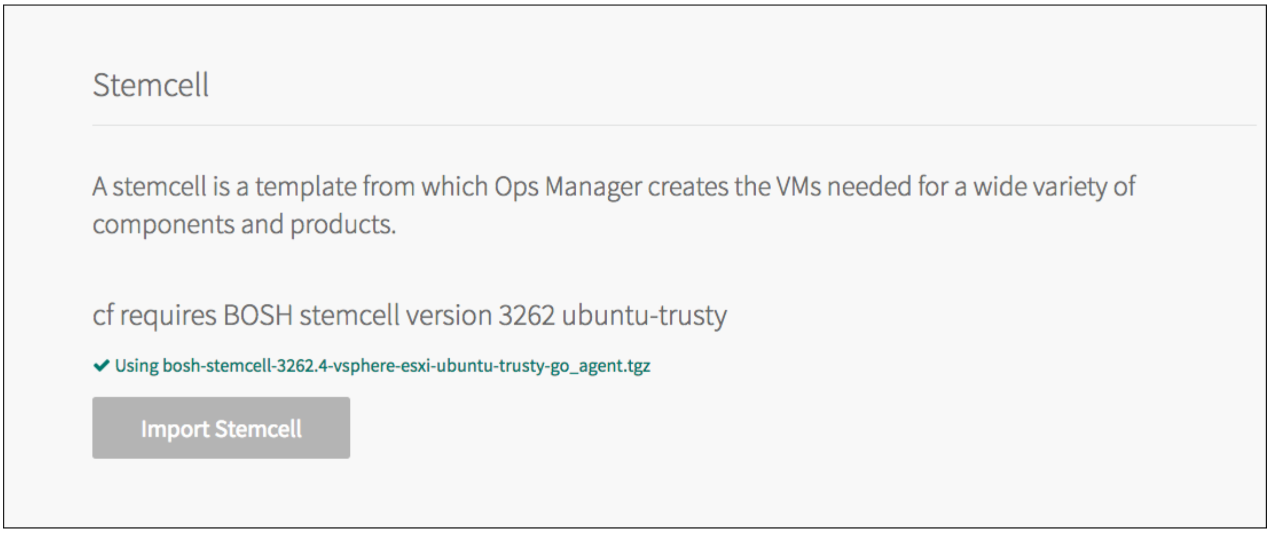
This step is only required if your Ops Manager does not already have the stemcell version required by PAS.

1. Select **Stemcell**.

2. Log into the Pivotal Network and click on **Stemcells**.

3. Download the appropriate stemcell version targeted for your IaaS.

4. In Ops Manager, import the downloaded stemcell `.tgz` file.

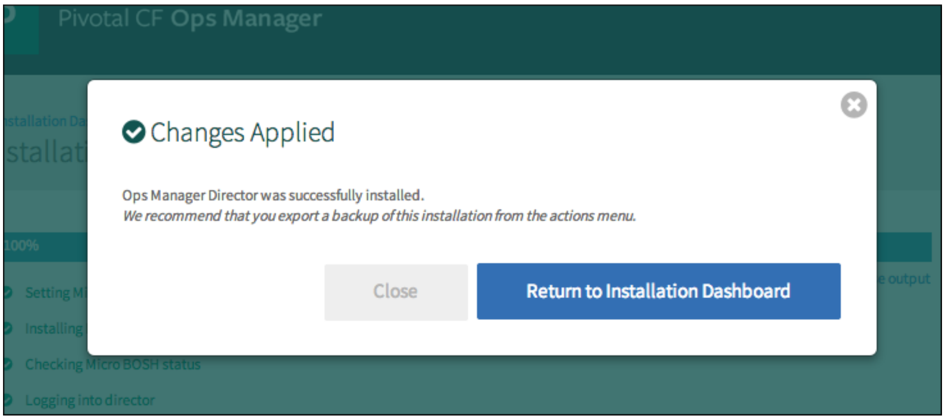


Step 24: Complete the PAS Installation

1. Click the **Installation Dashboard** link to return to the Installation Dashboard.

2. Click **Apply Changes** to begin your installation of PAS.

   The install process generally requires a minimum of 90 minutes to complete. The image shows the **Changes Applied** window that displays when the installation process successfully completes.



Installing PCF on AWS using Terraform

Complete the following procedures to install Pivotal Cloud Foundry (PCF) on Amazon Web Services (AWS) using Terraform:

1. Preparing to Deploy PCF on AWS (Terraform)
2. Configuring Ops Manager Director on AWS (Terraform)
3. (Optional) Installing the PCF IPsec Add-On
4. Deploying PAS on AWS (Terraform)

Preparing to Deploy PCF on AWS (Terraform)

Page last updated:

This guide describes the preparation steps required to install Pivotal Cloud Foundry (PCF) on Amazon Web Services (AWS) using Terraform templates.

The Terraform template for PCF on AWS describes a set of AWS resources and properties. For more information about how Terraform creates resources in AWS, see the AWS Provider topic on the Terraform site.

You may also find it helpful to review different deployment options in the Reference Architecture for Pivotal Cloud Foundry on AWS.

Prerequisites

In addition to fulfilling the prerequisites listed in the Installing Pivotal Cloud Foundry on AWS topic, ensure you have the following:

- The Terraform CLI
- In your AWS project, ensure you have an IAM user with the following permissions:
  - AmazonEC2FullAccess
  - AmazonRDSFullAccess
  - AmazonRoute53FullAccess
  - AmazonS3FullAccess
  - AmazonVPCFullAccess
  - IAMFullAccess

Step 1: Download and Edit the Terraform Variables File

Before you can run Terraform commands to create infrastructure resources, you must fill out a template variables file.

1. Navigate to the Pivotal Application Service (PAS) release on Pivotal Network.

2. Download the AWS Terraform zip file.

3. Extract the contents of the zip file and place the folder in your `workspace` directory on your local machine.

4. From a terminal window, navigate to the folder:

```
$ cd ~/workspace/TERRAFORMING-AWS-FOLDER
```

5. Create a new file named `terraform.tfvars`.

```
$ touch terraform.tfvars
```

6. Open the `terraform.tfvars` file and paste in the following contents:

```
env_name           = "YOUR-ENVIRONMENT-NAME"
access_key         = "YOUR-ACCESS-KEY"
secret_key         = "YOUR-SECRET-KEY"
region             = "YOUR-AWS-REGION"
availability_zones = ["YOUR-AZ-1", "YOUR-AZ-2", "YOUR-AZ-3"]
ops_manager_ami    = "YOUR-OPS-MAN-IMAGE-AMI"
dns_suffix         = "YOUR-DNS-SUFFIX"

ssl_cert = <<SSL_CERT
-----BEGIN CERTIFICATE-----
YOUR-CERTIFICATE
-----END CERTIFICATE-----
SSL_CERT

ssl_private_key = <<SSL_KEY
-----BEGIN EXAMPLE RSA PRIVATE KEY-----
YOUR-PRIVATE-KEY
-----END EXAMPLE RSA PRIVATE KEY-----
SSL_KEY
```

7. Edit the values in the file according to the table below:

Value to replace	Guidance
`YOUR-ENVIRONMENT-NAME`	Enter a name to use to identify resources in AWS. Terraform prepends the names of the resources it creates with this environment name. Example: `pcf`.
`YOUR-ACCESS-KEY`	Enter your AWS Access Key ID of the AWS project in which you want Terraform to create resources.
`YOUR-SECRET-KEY`	Enter your AWS Secret Access Key of the AWS project in which you want Terraform to create resources.
`YOUR-AWS-REGION`	Enter the name of the AWS region in which you want Terraform to create resources. Example: `us-central1`.

`YOUR-AZ-1`
`YOUR-AZ-2`
`YOUR-AZ-3`
	Enter three availability zones from your region. Example: `us-central-1a`, `us-central-1b`, `us-central-1c`.

`YOUR-OPS-MAN-IMAGE-AMI`	Enter the source code for the Ops Manager image you want to boot. You can find this code in the PDF included with the Ops Manager release on Pivotal Network.
`YOUR-DNS-SUFFIX`	Enter a domain name to use as part of the system domain for your PCF deployment. Terraform creates DNS records in AWS using `YOUR-ENVIRONMENT-NAME` and `YOUR-DNS-SUFFIX`. For example, if you enter `example.com` for your DNS suffix and have `pcf` as your environment name, Terraform will create DNS records at `pcf.example.com`.
`YOUR-CERTIFICATE`	Enter a certificate to use for HTTP load balancing. For production environments, use a certificate from a Certificate Authority (CA). For test environments, you can use a self-signed certificate. Your certificate must specify your system domain as the common name. Your system domain is `YOUR-ENVIRONMENT-NAME.YOUR-DNS-SUFFIX`. It also must include the following subdomains: `*.sys.YOUR-SYSTEM-DOMAIN`, `*.login.sys.YOUR-SYSTEM-DOMAIN`, `*.uaa.sys.YOUR-SYSTEM-DOMAIN`, `*.apps.YOUR-SYSTEM-DOMAIN`.
`YOUR-PRIVATE-KEY`	Enter a private key for the certificate you entered.

Step 2: Add Optional Variables

Complete this step if you want to do any of the following:

- Use an RDS for your deployment
- Deploy the Isolation Segment tile

In your `terraform.tfvars` file, specify the appropriate variables from the sections below.

Note: You can see the configurable options by opening the `variables.tf` file and looking for variables with default values.

Isolation Segments

If you plan to deploy the Isolation Segment tile, add the following variables to your `terraform.tfvars` file, replacing `YOUR-CERTIFICATE` and `YOUR-PRIVATE-KEY` with a certificate and private key. This causes terraform to create an additional HTTP load balancer across three availability zones to use for the Isolation Segment tile.


```
create_isoseg_resources = 1

iso_seg_ssl_cert = <<ISO_SEG_SSL_CERT
-----BEGIN CERTIFICATE-----
YOUR-CERTIFICATE
-----END CERTIFICATE-----
ISO_SEG_SSL_CERT

iso_seg_ssl_cert_private_key = <<ISO_SEG_SSL_KEY
-----BEGIN EXAMPLE RSA PRIVATE KEY-----
YOUR-PRIVATE-KEY
-----END EXAMPLE RSA PRIVATE KEY-----
ISO_SEG_SSL_KEY
```

RDS

1. If you want to use an RDS for Ops Manager and PAS, add the following to your `terraform.tfvars` file:

```
rds_instance_count = 1
```

2. If you want to specify a username for RDS authentication, add the following variable to your `terraform.tfvars` file.

```
rds_db_username = username
```

Step 3: Create AWS Resources with Terraform

Follow these steps to use the Terraform CLI to create resources on AWS:

1. From the directory that contains the Terraform files, run `terraform init` to initialize the directory based on the information you specified in the `terraform.tfvars` file.

```
$ terraform init
```

2. Run `terraform plan -out=plan` to create the execution plan for Terraform.

```
$ terraform plan -out=plan
```

3. Run `terraform apply plan` to execute the plan from the previous step. It may take several minutes for Terraform to create all the resources in AWS.

```
$ terraform apply plan
```

Step 5: Create DNS Record

1. In a browser, navigate to the DNS provider for the DNS suffix you entered in your `terraform.tfvars` file.

2. Create a new NS (Name server) record for your PCF system domain. Your system domain is `YOUR-ENVIRONMENT-NAME.YOUR-DNS-SUFFIX`.

3. In this record, enter the name servers included in `env_dns_zone_name_servers` from your Terraform output.

What to Do Next

Proceed to the next step in the deployment, Configuring Ops Manager Director on AWS (Terraform).

Configuring Ops Manager Director on AWS (Terraform)

Page last updated:

This topic describes how to configure the Ops Manager Director for Pivotal Cloud Foundry (PCF) on Amazon Web Services (AWS) after Preparing to Deploy PCF on AWS (Terraform).

> **Note**: You can also perform the procedures in this topic using the Ops Manager API. For more information, see the Using the Ops Manager API topic.

Prerequisite

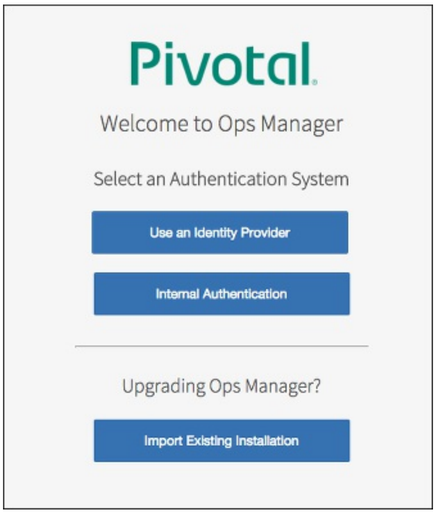
To complete the procedures in this topic, you must have access to the output from when you ran `terraform apply` to create resources for this deployment. You can view this output at any time by running `terraform output`. You use the values in your Terraform output to configure the Ops Manager Director tile.

Step 1: Access Ops Manager

> **Note**: For security, Ops Manager v1.7 and later require that you log in using a fully qualified domain name.

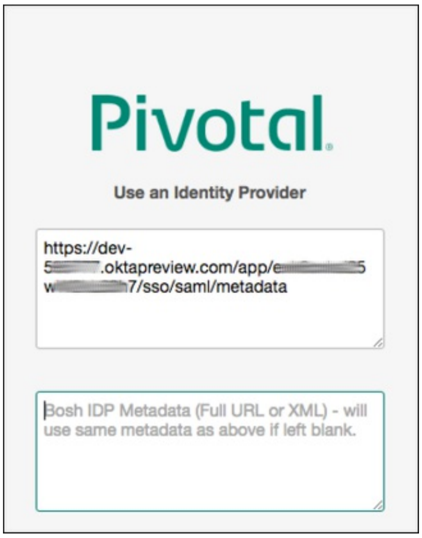
1. In a web browser, navigate to the fully qualified domain name (FQDN) of the Ops Manager director. Use the `ops_manager_dns` value from running `terraform output`.

2. When Ops Manager starts for the first time, you must choose one of the following:
    - Use an Identity Provider: If you use an Identity Provider, an external identity server maintains your user database.
    - Internal Authentication: If you use Internal Authentication, PCF maintains your user database.



Use an Identity Provider (IdP)

1. Log in to your IdP console and download the IdP metadata XML. Optionally, if your IdP supports metadata URL, you can copy the metadata URL instead of the XML.



2. Copy the IdP metadata XML or URL to the Ops Manager **Use an Identity Provider** log in page.

   > **Note**: The same IdP metadata URL or XML is applied for the BOSH Director. If you use a separate IdP for BOSH, copy the metadata XML or URL from that IdP and enter it into the BOSH IdP Metadata text box in the Ops Manager log in page.

3. Enter your **Decryption passphrase**. Read the **End User License Agreement**, and select the checkbox to accept the terms.

4. Your Ops Manager log in page appears. Enter your username and password. Click **Login**.

5. Download your SAML Service Provider metadata (SAML Relying Party metadata) by navigating to the following URLs:
   - **5a.** Ops Manager SAML service provider metadata: `https://OPS-MAN-FQDN:443/uaa/saml/metadata`
   - **5b.** BOSH Director SAML service provider metadata: `https://BOSH-IP-ADDRESS:8443/saml/metadata`

   > **Note**: To retrieve your `BOSH-IP-ADDRESS`, navigate to the **Ops Manager Director** tile > **Status** tab. Record the **Ops Manager Director** IP address.

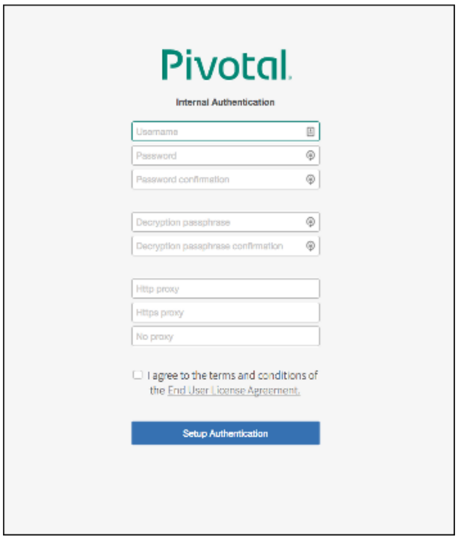
6. Configure your IdP with your SAML Service Provider metadata. Import the Ops Manager SAML provider metadata from Step 5a above to your IdP. If your IdP does not support importing, provide the values below.
   - **Single sign on** URL: `https://OPS-MAN-FQDN:443/uaa/saml/SSO/alias/OPS-MAN-FQDN`
   - **Audience URI** (SP Entity ID): `https://OP-MAN-FQDN:443/uaa`
   - **Name ID**: Email Address
   - SAML authentication requests are always signed

7. Import the BOSH Director SAML provider metadata from Step 5b to your IdP. If the IdP does not support an import, provide the values below.
   - **Single sign on** URL: `https://BOSH-IP:8443/saml/SSO/alias/BOSH-IP`
   - **Audience URI** (SP Entity ID): `https://BOSH-IP:8443`
   - **Name ID**: Email Address
   - SAML authentication requests are always signed

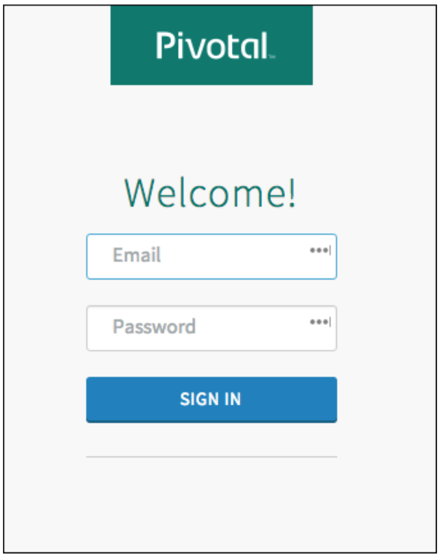
8. Return to the **Ops Manager Director** tile, and continue with the configuration steps below.

Internal Authentication

1. When redirected to the **Internal Authentication** page, you must complete the following steps:
   - Enter a **Username**, **Password**, and **Password confirmation** to create an Admin user.
   - Enter a **Decryption passphrase** and the **Decryption passphrase confirmation**. This passphrase encrypts the Ops Manager datastore, and is not recoverable if lost.
   - If you are using an **HTTP proxy** or **HTTPS proxy**, follow the instructions in the Configuring Proxy Settings for the BOSH CPI topic.
   - Read the **End User License Agreement**, and select the checkbox to accept the terms.
   - Click **Setup Authentication**.



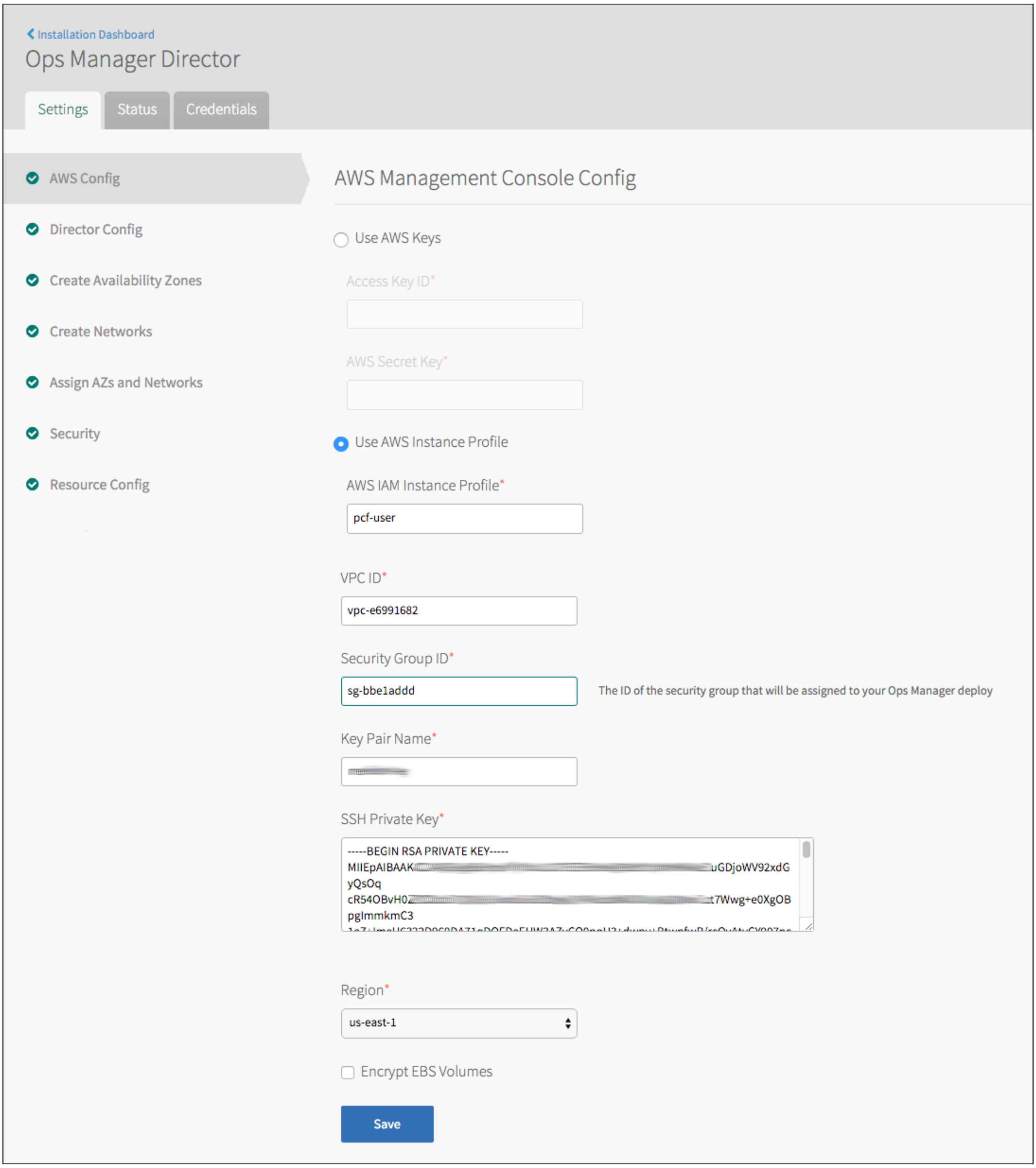
2. Log in to Ops Manager with the Admin username and password that you created in the previous step.



Step 2: AWS Config Page

1. Click the **Ops Manager Director** tile.

2. Select **AWS Config** to open the **AWS Management Console Config** page.



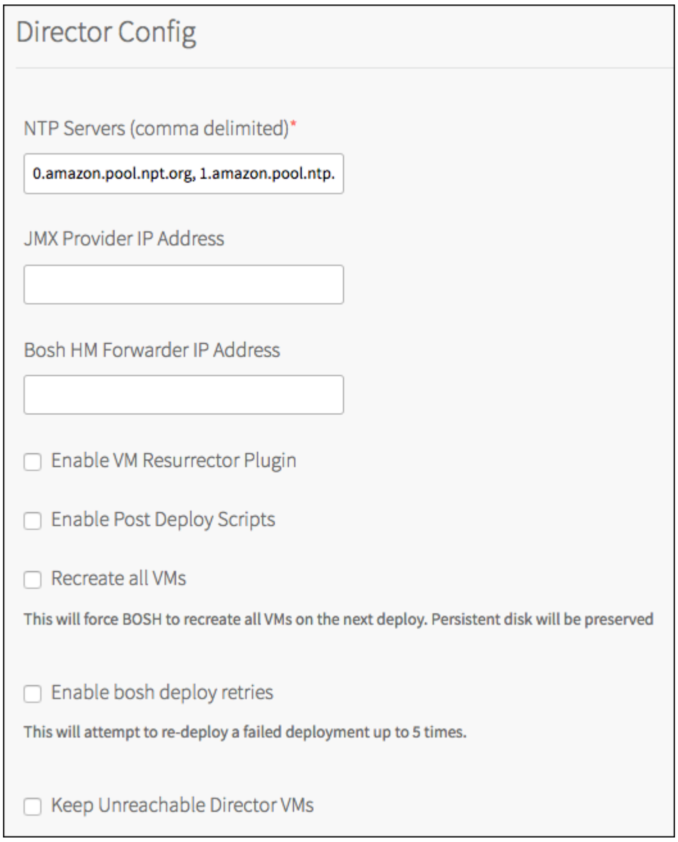
3. Select **Use AWS Keys** or **Use AWS Instance Profile**.
   - If you choose to use AWS keys, complete the following fields:
     - **Access Key ID**: Enter the value of `iam_user_access_key` from the Terraform output.
     - **AWS Secret Key**: Enter the value of `iam_user_secret_access_key` from the Terraform output.
   - If you choose to use an AWS instance profile, enter the name of your AWS Identity and Access Management (IAM) profile.
4. Complete the remainder of the **AWS Management Console Config** page with the following information.
   - **VPC ID**: Enter the value of `vpc_id` from the Terraform output.
   - **Security Group ID**: Enter the value of `vms_security_group_id` from the Terraform output.
   - **Key Pair Name**: Enter the value of `ops_manager_ssh_public_key_name` from the Terraform output.
   - **SSH Private Key**: Enter the value of `ops_manager_ssh_private_key` from the Terraform output.
   - **Region**: Select the region where you deployed Ops Manager.
   - **Encrypt EBS Volumes**: Select this checkbox to enable full encryption on persistent disks of all BOSH-deployed virtual machines (VMs), except for the Ops Manager VM and Director VM. See the Configuring Amazon EBS Encryption for PCF on AWS topic for details about using EBS

encryption.

5. Click **Save**.

Step 3: Director Config Page

1. Select **Director Config** to open the **Director Config** page.



2. Enter at least two of the following NTP servers in the **NTP Servers (comma delimited)** field, separated by a comma: `0.amazon.pool.ntp.org,1.amazon.pool.ntp.org,2.amazon.pool.ntp.org,3.amazon.pool.ntp.org`

3. Leave the **JMX Provider IP Address** field blank.

> **Note**: Starting from PCF v2.0, BOSH-reported system metrics are available in the Loggregator Firehose by default. Therefore, if you continue to use PCF JMX Bridge for consuming them outside of the Firehose, you may receive duplicate data. To prevent this, leave the **JMX Provider IP Address** field blank. For additional guidance, see BOSH System Metrics Available in Loggregator Firehose.

4. Leave the **Bosh HM Forwarder IP Address** field blank.

> **Note**: Starting from PCF v2.0, BOSH-reported system metrics are available in the Loggregator Firehose by default. Therefore, if you continue to use the BOSH HM Forwarder for consuming them, you may receive duplicate data. To prevent this, leave the **Bosh HM Forwarder IP Address** field blank. For additional guidance, see BOSH System Metrics Available in Loggregator Firehose.

5. Select the **Enable VM Resurrector Plugin** checkbox to enable the Ops Manager Resurrector functionality and increase Pivotal Application Service (PAS) availability.

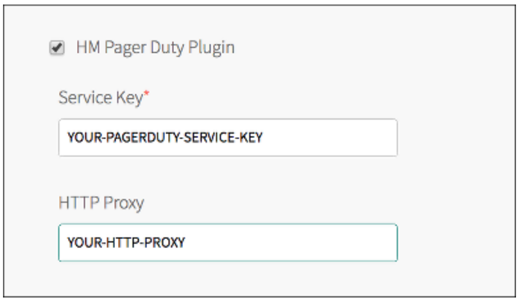
6. Select **Enable Post Deploy Scripts** to run a post-deploy script after deployment. This script allows the job to execute additional commands against a deployment.

7. Select **Recreate all VMs** to force BOSH to recreate all VMs on the next deploy. This process does not destroy any persistent disk data.

8. Select **Enable bosh deploy retries** if you want Ops Manager to retry failed BOSH operations up to five times.

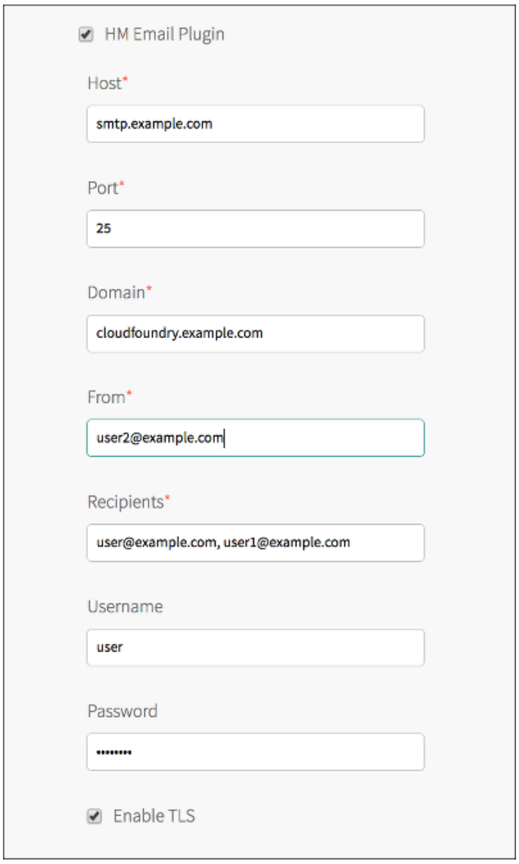
9. Select **Keep Unreachable Director VMs** if you want to preserve Ops Manager Director VMs after a failed deployment for troubleshooting purposes.

10. Select **HM Pager Duty Plugin** to enable Health Monitor integration with PagerDuty.



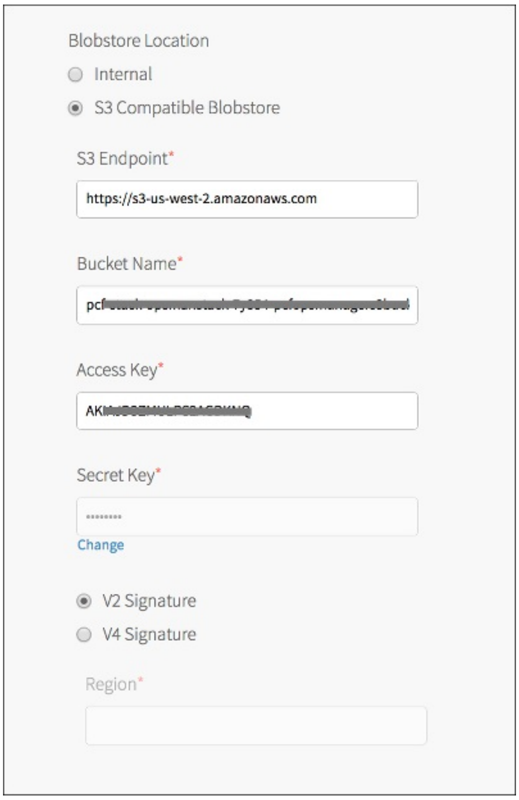
    - **Service Key**: Enter your API service key from PagerDuty.
    - **HTTP Proxy**: Enter an HTTP proxy for use with PagerDuty.

11. Select **HM Email Plugin** to enable Health Monitor integration with email.



    - **Host**: Enter your email hostname.
    - **Port**: Enter your email port number.
    - **Domain**: Enter your domain.
    - **From**: Enter the address for the sender.
    - **Recipients**: Enter comma-separated addresses of intended recipients.
    - **Username**: Enter the username for your email server.
    - **Password**: Enter the password password for your email server.
    - **Enable TLS**: Select this checkbox to enable Transport Layer Security.

12. For **Blobstore Location**, select **S3 Compatible Blobstore** and complete the following steps:



a. In a browser, reference the Amazon Simple Storage Service (Amazon S3) table, and find the region for your AWS account.
b. Prepend `https://` to the **Endpoint** for your region, and copy it into the Ops Manager **S3 Endpoint** field. For example, in the **us-west-2** region, enter `https://s3-us-west-2.amazonaws.com` into the field.
c. Complete the following fields:
   - **Bucket Name**: Enter the value of `ops_manager_bucket` from the Terraform output.
   - **Access Key ID**: Enter the value of `iam_user_access_key` from the Terraform output.
   - **AWS Secret Key**: Enter the value of `iam_user_secret_access_key` from the Terraform output.

d. Select **V2 Signature** or **V4 Signature**. If you select **V4 Signature**, enter your **Region**.

> **Note**: AWS recommends using Signature Version 4 when possible. For more information about AWS S3 Signatures, see the AWS Authenticating Requests documentation.

13. For **Database Location**, select **Internal**.

14. (Optional) **Director Workers** sets the number of workers available to execute Director tasks. This field defaults to `5`.

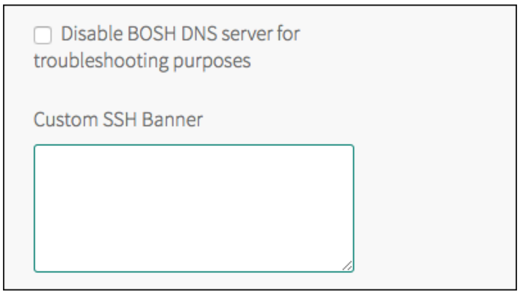
15. (Optional) **Max Threads** sets the maximum number of threads that the Ops Manager Director can run simultaneously. Pivotal recommends that you leave the field blank to use the default value, unless doing so results in rate limiting or errors on your IaaS.

16. (Optional) To add a custom URL for your Ops Manager Director, enter a valid hostname in **Director Hostname**. You can also use this field to configure a load balancer in front of your Ops Manager Director. For more information, see How to set up a load balancer in front of Ops Manager Director in the Pivotal Knowledge Base.

17. (Optional) To disable BOSH DNS, select the **Disable BOSH DNS server for troubleshooting purposes** checkbox. For more information about the BOSH DNS service discovery mechanism, see BOSH DNS Service Discovery for Application Containers in the PAS Release Notes.

> **Breaking Change**: Do not disable BOSH DNS without consulting Pivotal Support. For more information about disabling BOSH DNS, see Disabling or Opting Out of BOSH DNS in PCF in the Pivotal Knowledge Base.

18. (Optional) To set a custom banner that users see when logging in to the Director using SSH, enter text in the **Custom SSH Banner** field.

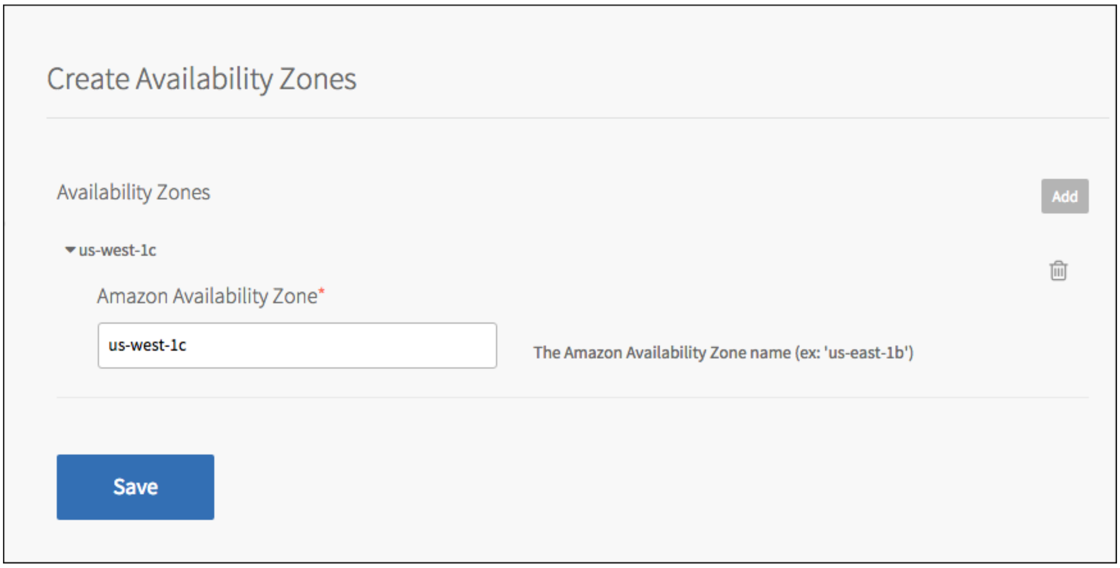


19. Click **Save**.

Note: For more information about AWS S3 Signatures, see the AWS [Authenticating Requests](#) documentation.

Step 4: Create Availability Zones Page

1. Select **Create Availability Zones**.

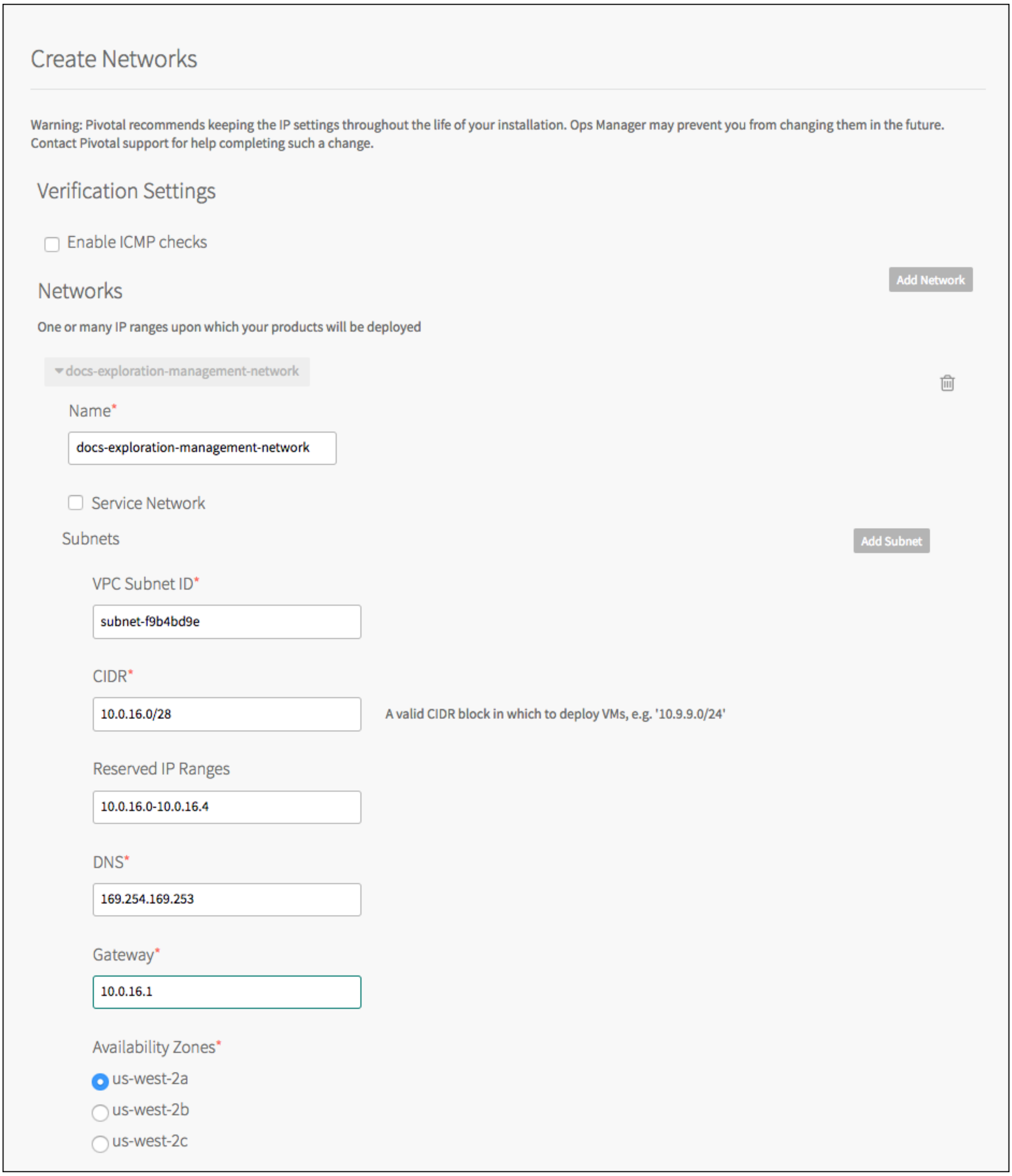


2. Use the following steps to create three Availability Zones for your apps to use:

   a. Click **Add** three times.
   b. For **Amazon Availability Zone**, enter the values corresponding to the key `management_subnet_availability_zones` from the Terraform output.
   c. Click **Save**.

Step 5: Create Networks Page

1. Select **Create Networks**.



2. (Optional) Select **Enable ICMP checks** to enable ICMP on your networks. Ops Manager uses ICMP checks to confirm that components within your network are reachable.

3. Perform the following steps to add the network configuration you created for your VPC:

   a. Click **Add Network**.
   b. For **Name**, enter `pcf-management-network`.
   c. Create a subnet for each availability zone by clicking **Add Subnet**. Refer to the table below for the information required to create all three subnets:

VPC Subnet ID	CIDR	Reserved IP Ranges	DNS	Gateway	Availability Zones
The first value of `management_subnet_ids` from the Terraform output.	`10.0.16.0/28`	`10.0.16.0-10.0.16.4`	`169.254.169.253`	`10.0.16.1`	The first value of `management_subnet_availability_zones` from the Terraform output.
The second value of					The second value of

`management_subnet_ids` from the Terraform output.	`10.0.16.16/28`	`10.0.16.16-10.0.16.20`	`169.254.169.253`	`10.0.16.17`	`management_subnet_availability_zones` from the Terraform output.
The third value of `management_subnet_ids` from the Terraform output.	`10.0.16.32/28`	`10.0.16.32-10.0.16.36`	`169.254.169.253`	`10.0.16.33`	The third value of `management_subnet_availability_zones` from the Terraform output.

d. Click **Add Network**.
e. For **Name**, enter `pcf-pas-network`.
f. Create a subnet for each availability zone by clicking **Add Subnet**. Refer to the table below for the information required to create all three subnets:

VPC Subnet ID	CIDR	Reserved IP Ranges	DNS	Gateway	Availability Zones
The first value of `pas_subnet_ids` from the Terraform output.	`10.0.4.0/24`	`10.0.4.0-10.0.4.4`	`169.254.169.253`	`10.0.4.1`	The first value of `pas_subnet_availability_zones` from the Terraform output.
The second value of `pas_subnet_ids` from the Terraform output.	`10.0.5.0/24`	`10.0.5.0-10.0.5.4`	`169.254.169.253`	`10.0.5.1`	The second value of `pas_subnet_availability_zones` from the Terraform output.
The third value of `pas_subnet_ids` from the Terraform output.	`10.0.6.0/24`	`10.0.6.0-10.0.6.4`	`169.254.169.253`	`10.0.6.1`	The third value of `pas_subnet_availability_zones` from the Terraform output.

g. Click **Add Network**.
h. Check the **Service Network** box. This allows you to dynamically provision VMs in this network for use with on-demand services. Ops Manager does not provision VMs within the specified CIDR range.
i. Enter `pcf-services-network` in **Name** field.
j. Create a subnet for each availability zone by clicking **Add Subnet**. Refer to the table below for the information required to create all three subnets:

VPC Subnet ID	CIDR	Reserved IP Ranges	DNS	Gateway	Availability Zones
The first value of `services_subnet_ids` from the Terraform output.	`10.0.8.0/24`	`10.0.8.0-10.0.8.3`	`169.254.169.253`	`10.0.8.1`	The first value of `services_subnet_availability_zones` from the Terraform output.
The second value of `services_subnet_ids` from the Terraform output.	`10.0.9.0/24`	`10.0.9.0-10.0.9.3`	`169.254.169.253`	`10.0.9.1`	The second value of `services_subnet_availability_zones` from the Terraform output.
The third value of `services_subnet_ids` from the Terraform output.	`10.0.10.0/24`	`10.0.10.0-10.0.10.3`	`169.254.169.253`	`10.0.10.1`	The third value of `services_subnet_availability_zones` from the Terraform output.

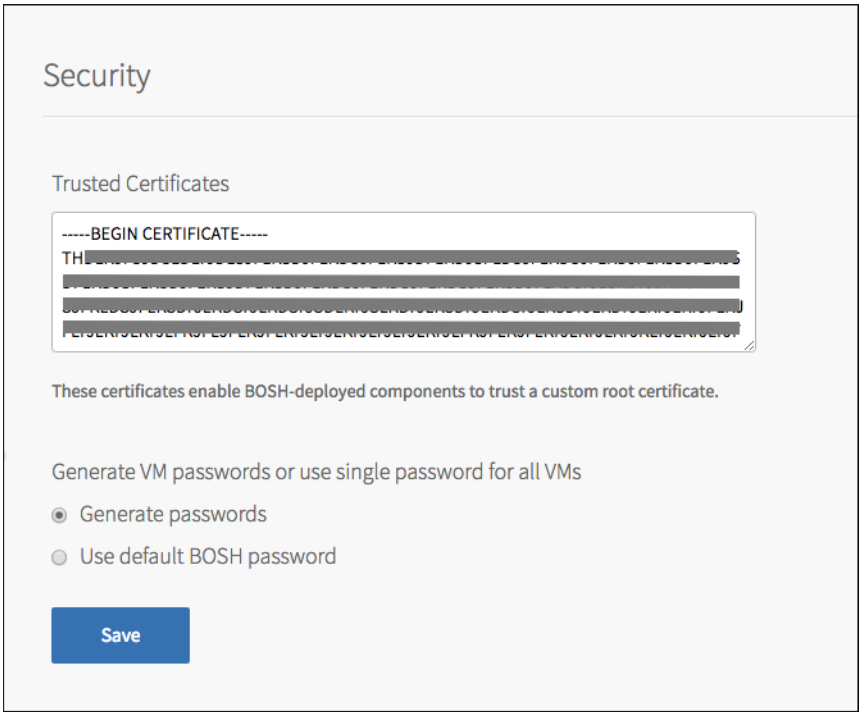
4. Click **Save**.

Step 6: Assign AZs and Networks Page

1. Select **Assign AZs and Networks**.
2. Use the drop-down menu to select a **Singleton Availability Zone**. The Ops Manager Director installs in this Availability Zone.
3. Use the drop-down menu to select `pcf-management-network` for your Ops Manager Director.
4. Click **Save**.

Step 7: Security Page

1. Select **Security**.



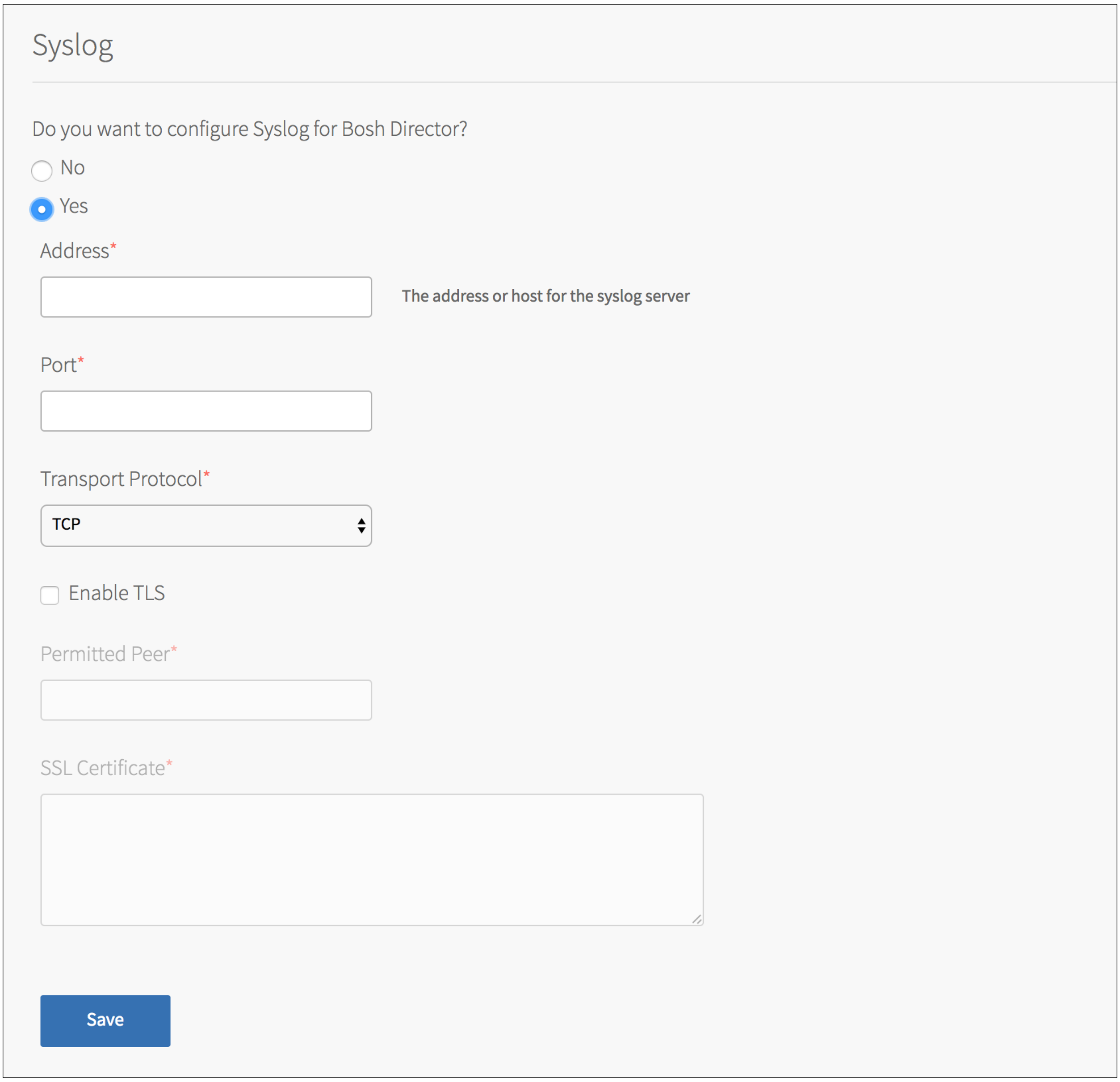
2. In **Trusted Certificates**, enter a custom certificate authority (CA) certificate to insert into your organization's certificate trust chain. This feature enables all BOSH-deployed components in your deployment to trust a custom root certificate. If you want to use Docker Registries for running app instances in Docker containers, use this field to enter your certificate for your private Docker Registry. See the Using Docker Registries topic for more information.

3. Choose **Generate passwords** or **Use default BOSH password**. Pivotal recommends that you use the **Generate passwords** option for greater security.

4. Click **Save**. To view your saved Director password, click the **Credentials** tab.

Step 8: Syslog Page

1. Select **Syslog**.



2. (Optional) To send BOSH Director system logs to a remote server, select **Yes**.

3. In the **Address** field, enter the IP address or DNS name for the remote server.

4. In the **Port** field, enter the port number that the remote server listens on.

5. In the **Transport Protocol** dropdown menu, select **TCP**, **UDP**, or **RELP**. This selection determines which transport protocol is used to send the logs to the remote server.

6. (Optional) Mark the **Enable TLS** checkbox to use TLS encryption when sending logs to the remote server.
   - In the **Permitted Peer** field, enter either the name or SHA1 fingerprint of the remote peer.
   - In the **SSL Certificate** field, enter the SSL certificate for the remote server.

7. Click **Save**.

Step 9: Resource Config Page

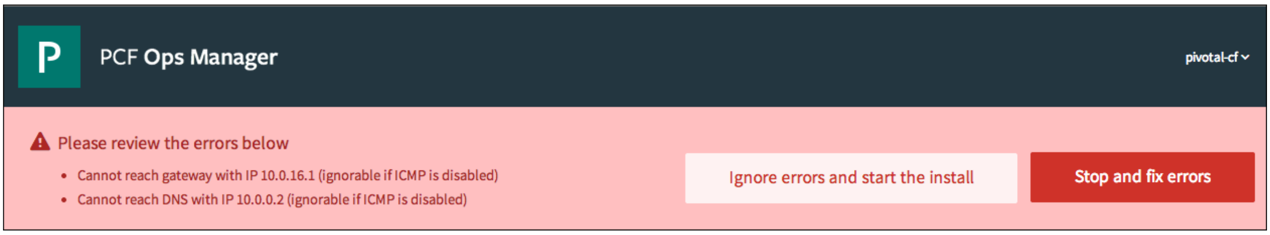
1. Select **Resource Config**.

2. Adjust any values as necessary for your deployment. Under the **Instances**, **Persistent Disk Type**, and **VM Type** fields, choose **Automatic** from the drop-down menu to allocate the recommended resources for the job. If the **Persistent Disk Type** field reads **None**, the job does not require persistent disk space.

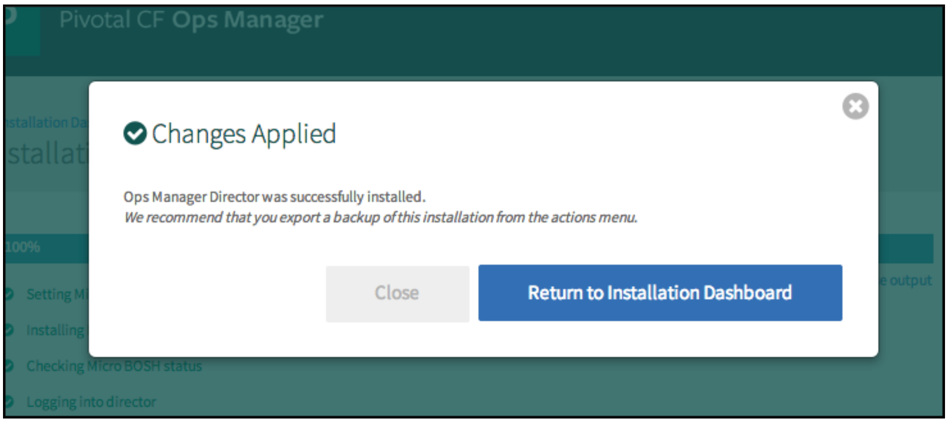
> **Note**: If you set a field to **Automatic** and the recommended resource allocation changes in a future version, Ops Manager automatically uses the updated recommended allocation.

Step 10: Complete the Ops Manager Director Installation

1. Click the **Installation Dashboard** link to return to the Installation Dashboard.
2. Click **Apply Changes**. If the following ICMP error message appears, click **Ignore errors and start the install**.



3. Ops Manager Director installs. This may take a few moments. When the installation process successfully completes, the **Changes Applied** window appears.



4. After you complete this procedure, follow the instructions in the Deploying PAS on AWS (Terraform) topic.

Deploying PAS on AWS (Terraform)

Page last updated:

This topic describes how to install and configure Pivotal Application Service (PAS) on Amazon Web Services (AWS).

Before beginning this procedure, ensure that you have successfully completed the Configuring Ops Manager Director on AWS (Terraform) topic.

Note: If you plan to install the PCF IPsec add-on, you must do so before installing any other tiles. Pivotal recommends installing IPsec immediately after Ops Manager, and before installing the PAS tile.

Step 1: Add PAS to Ops Manager

1. If you have not already downloaded PAS, log in to Pivotal Network, and click PAS.

2. From the **Releases** drop-down, select the release to install and choose one of the following:

    a. Click PAS to download the PAS `.pivotal` file
    b. Click **PCF Small Footprint Runtime** to download the Small Footprint Runtime `.pivotal` file. For more information, see Getting Started with Small Footprint Runtime.

3. Navigate to the Pivotal Cloud Foundry Operations Manager Installation Dashboard.

4. Click **Import a Product** to add your tile to Ops Manager. For more information, refer to the Adding and Deleting Products topic.

5. Click the PAS tile in the Installation Dashboard.

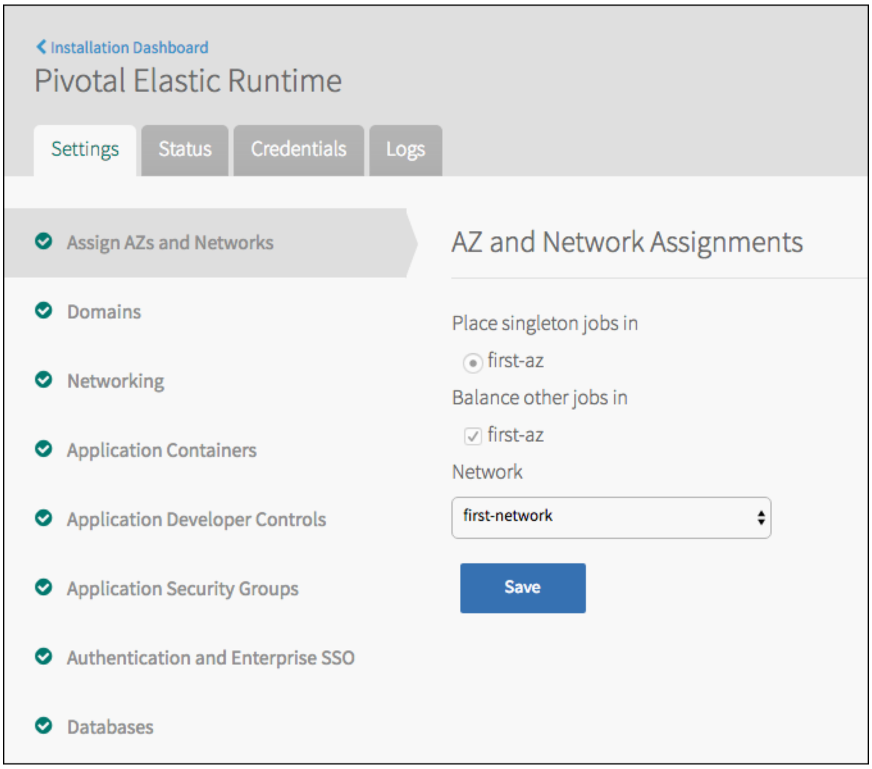
Step 2: Assign Availability Zones and Networks

1. Select **Assign AZ and Networks**.

2. Select an Availability Zone under **Place singleton jobs**. Ops Manager runs any job with a single instance in this Availability Zone.

3. Select all three Availability Zones under **Balance other jobs**. Ops Manager balances instances of jobs with more than one instance across the Availability Zones that you specify.

4. From the **Network** drop-down box, choose the `pcf-pas-network` you created when configuring the Ops Manager Director tile.

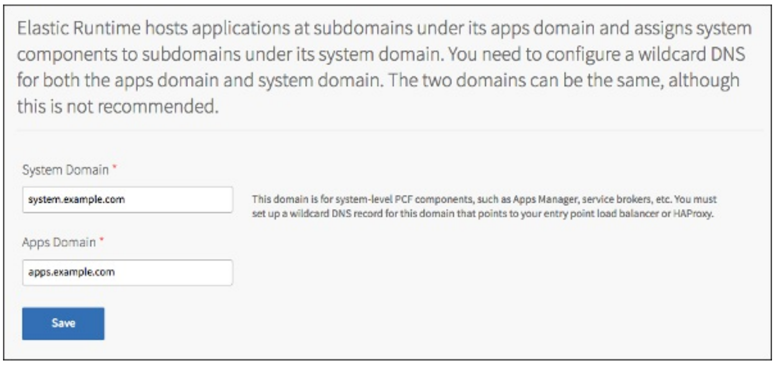


5. Click **Save**.

Note: When you save this form, a verification error displays because the PCF security group blocks ICMP. You can ignore this error.

Step 3: Configure Domains

1. Select **Domains**.



2. Enter the system and application domains.
   - For **System Domain**, enter the value of `sys_domain` from the Terraform output. This defines your target when you push apps to PAS.
   - For **Apps Domain**, enter the value of `app_domain` from the Terraform output. This defines where PAS should serve your apps.

Note: You configured a wildcard DNS record for these domains in an earlier step.

3. Click **Save**.

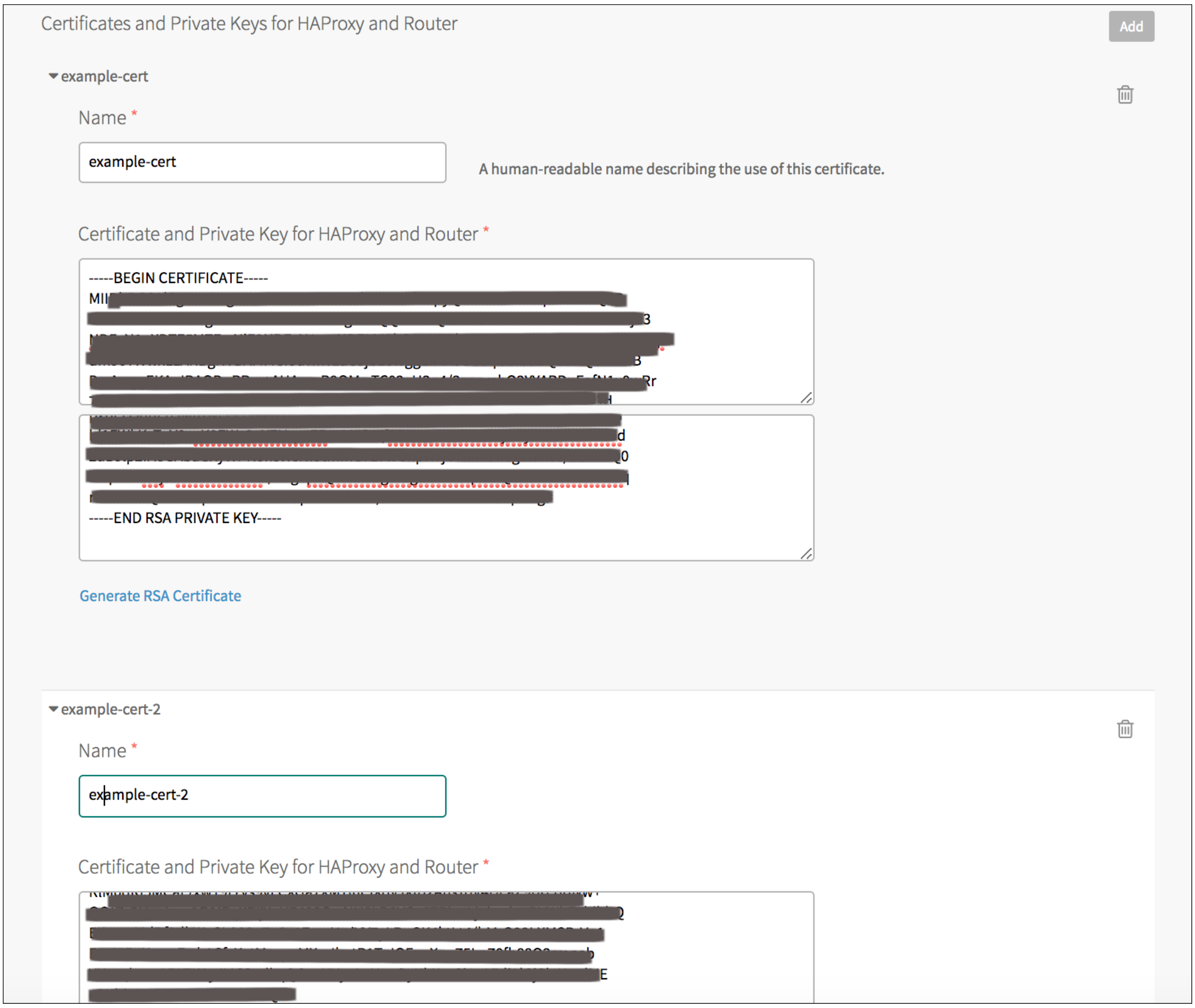
Step 4: Configure Networking

1. Select **Networking**.

2. Leave the **Router IPs**, **SSH Proxy IPs**, **HAProxy IPs**, and **TCP Router IPs** fields blank. You do not need to complete these fields when deploying PCF to AWS with Elastic Load Balancers (ELBs).

   > **Note**: You specify load balancers in the **Resource Config** section of Pivotal Application Service (PAS) later in the installation process. See the Configure Router to Elastic Load Balancer section of this topic for more information.

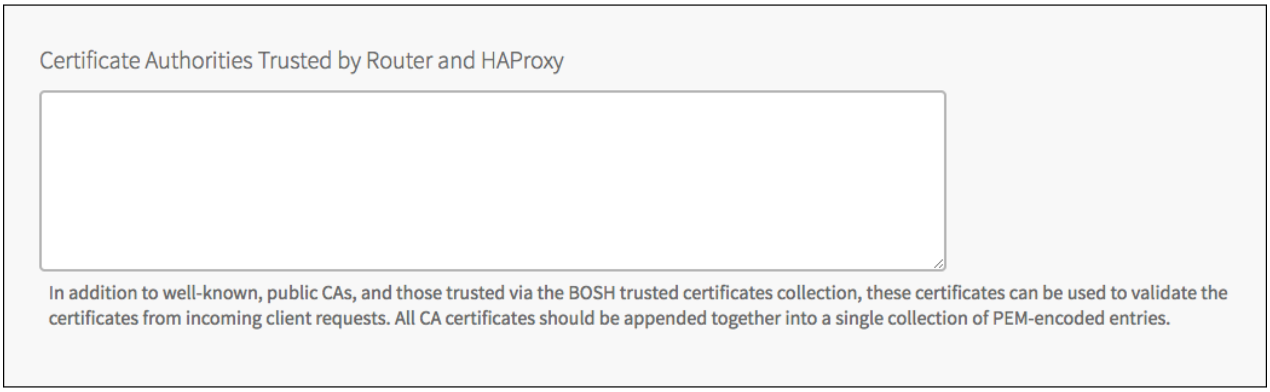
3. Under **Certificates and Private Key for HAProxy and Router**, you must provide at least one **Certificate and Private Key** name and certificate keypair for HAProxy and Gorouter. The HAProxy and Gorouter are enabled to receive TLS communication by default. You can configure multiple certificates for HAProxy and Gorouter. 1. Click the **Add** button to add a name for the certificate chain and its private keypair. This certificate is the default used by Gorouter and HAProxy.



   You can either provide a certificate signed by a Certificate Authority (CA) or click on the **Generate RSA Certificate** link to generate a self-signed certificate in Ops Manager. 1. If you want to configure multiple certificates for HAProxy and Gorouter, click the **Add** button and fill in the appropriate fields for each additional certificate keypair.

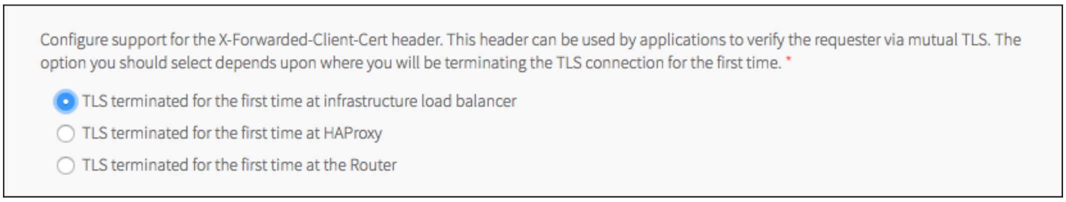
   For details about generating certificates in Ops Manager for your wildcard system domains, see the Providing a Certificate for Your SSL/TLS Termination Point topic.

4. (Optional) When validating client requests using mutual TLS, the Gorouter trusts multiple certificate authorities (CAs) by default. If you want to configure the Gorouter and HAProxy to trust additional CAs, enter your CA certificates under **Certificate Authorities Trusted by Router and HAProxy**. All CA certificates should be appended together into a single collection of PEM-encoded entries.



5. In the **Minimum version of TLS supported by HAProxy and Router** field, select the minimum version of TLS to use in HAProxy and Router communications. HAProxy and Router use TLS v1.2 by default. If you need to accommodate clients that use an older version of TLS, select a lower minimum version. For a list of TLS ciphers supported by the Gorouter, see [Securing Traffic into Cloud Foundry](#).

6. Under **Configure support for the X-Forwarded-Client-Cert header**, configure PCF handles `x-forwarded-client-cert` (XFCC) HTTP headers based on where TLS is terminated for the first time in your deployment.



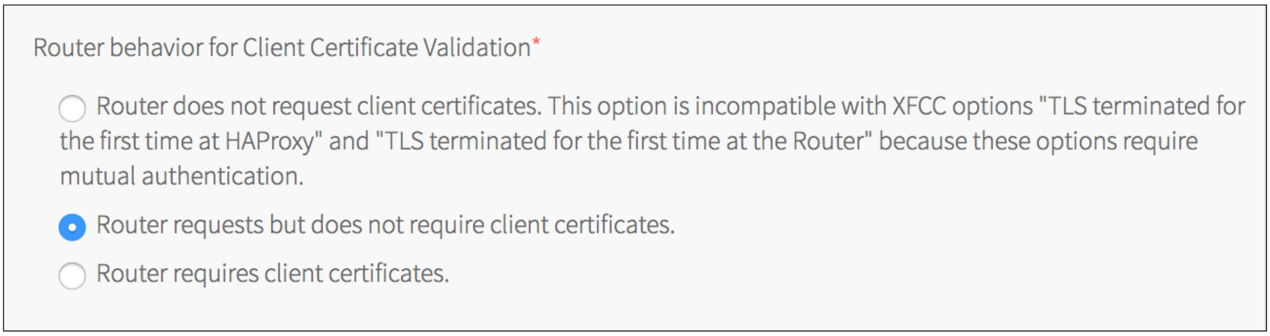
The following table indicates which option to choose based on your deployment layout.

If your deployment is configured as follows:	Then select the following option:	Additional notes:
◦ The Load Balancer is terminating TLS, and ◦ Load balancer is configured to put the client certificate from a mutual authentication TLS handshake into the X-Forwarded-Client-Cert HTTP header	TLS terminated for the first time at infrastructure load balancer (default).	Both HAProxy and the Gorouter forward the XFCC header when included in the request.
◦ The Load Balancer is configured to pass through the TLS handshake via TCP to the instances of HAProxy, and ◦ HAProxy instance count is > 0	TLS terminated for the first time at HAProxy.	HAProxy sets the XFCC header with the client certificate received in the TLS handshake. The Gorouter forwards the header. **Breaking Change**: In the **Router behavior for Client Certificates** field, you cannot select the **Router does not request client certificates** option.
◦ The Load Balancer is configured to pass through the TLS handshake via TCP to instances of the Gorouter	TLS terminated for the first time at the Gorouter.	The Gorouter strips the XFCC header if it is included in the request and forwards the client certificate received in the TLS handshake in a new XFCC header. If you have deployed instances of HAProxy, app traffic bypasses those instances in this configuration. If you have also configured your load balancer to route requests for ssh directly to the Diego Brain, consider reducing HAProxy instances to 0.

		Breaking Change: In the **Router behavior for Client Certificates** field, you cannot select the **Router does not request client certificates** option.

For a description of the behavior of each configuration option, see Forward Client Certificate to Applications.

7. To configure Gorouter behavior for handling client certificates, select one of the following options in the **Router behavior for Client Certificate Validation** field.



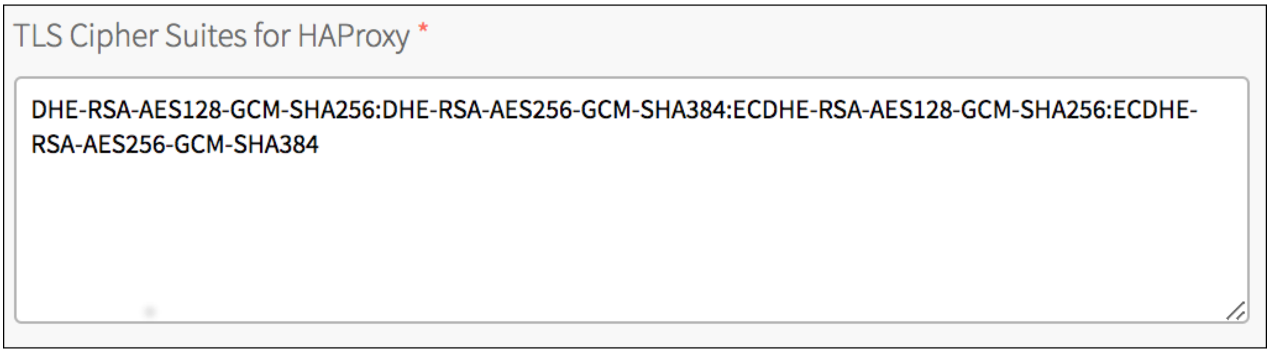
   - **Router does not request client certificates.** This option is incompatible with the XFCC configuration options **TLS terminated for the first time at HAProxy** and **TLS terminated for the first time at the Router** in PAS because these options require mutual authentication. As client certificates are not requested, client will not provide them, and thus validation of client certificates will not occur.
   - **Router requests but does not require client certificates.** Gorouter requests client certificates in TLS handshakes, validates them when presented, but does not require them. This is the default configuration.
   - **Router requires client certificates.** Gorouter validates that the client certificate is signed by a Certificate Authority that Gorouter trusts. If Gorouter cannot validate the client certificate, the TLS handshake fails.

8. In the **TLS Cipher Suites for Router** field, specify the TLS cipher suites to use for TLS handshakes between the Gorouter and downstream clients like load balancers or HAProxy. Use an ordered, colon-delimited list of Golang-supported TLS cipher suites in the OpenSSL format. The recommended setting is `ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384`. Operators should verify that the ciphers are supported by any clients or downstream components that will initiate TLS handshakes with the Gorouter. For a list of TLS ciphers supported by the Gorouter, see Securing Traffic into Cloud Foundry.

   Verify that whatever client is participating in the TLS handshake with the Gorouter has at least one cipher suite in common with the Gorouter.

   > **Note**: Specify cipher suites that are supported by the versions configured in the **Minimum version of TLS supported by HAProxy and Router** field.

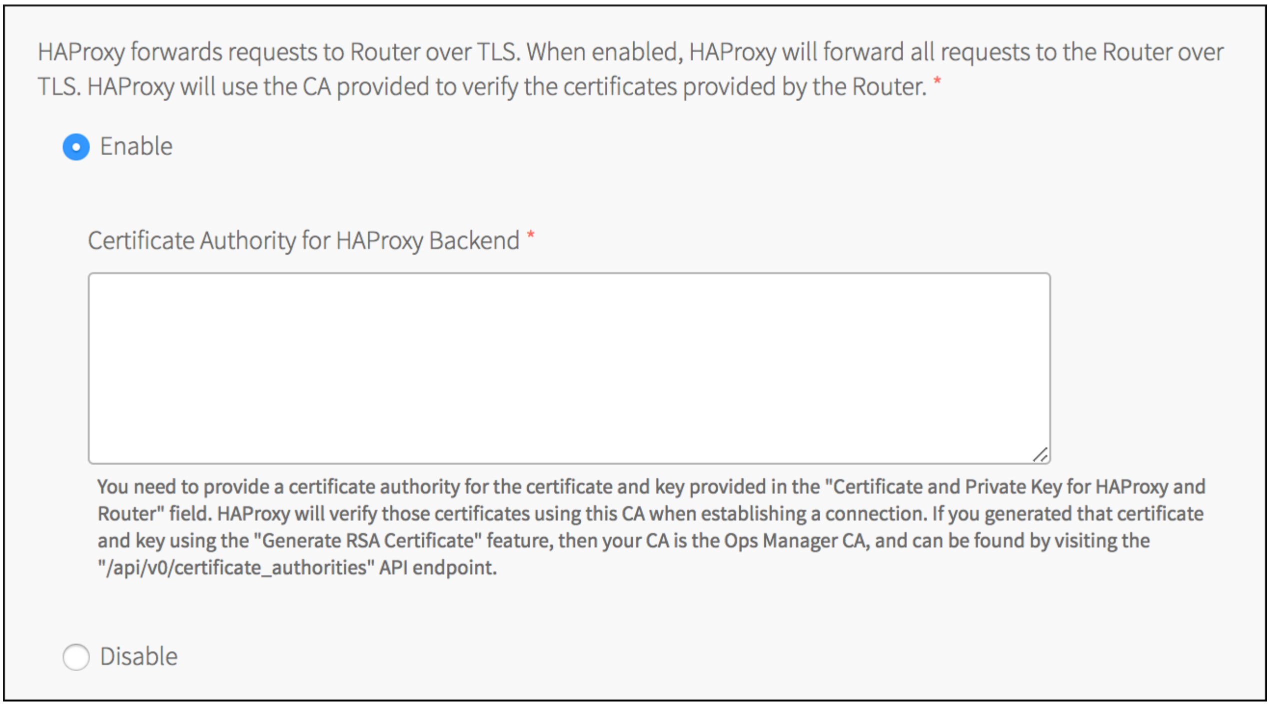
9. In the **TLS Cipher Suites for HAProxy** field, specify the TLS cipher suites to use in HAProxy for TLS handshakes between HAProxy and its clients such as load balancers and Gorouter. Use an ordered, colon-delimited list of TLS cipher suites in the OpenSSL format. The recommended setting: `DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384`
   Operators should verify that the ciphers are supported by any clients or downstream components that will initiate TLS handshakes with the HAProxy.



Verify that whatever clients are participating in the TLS handshake with HAProxy have at least one cipher suite in common with HAProxy.

> **Note:** Specify cipher suites that are supported by the versions configured in the **Minimum version of TLS supported by HAProxy and Router** field.

10. Under **HAProxy forwards requests to Router over TLS**, select **Enable** or **Disable** based on your deployment layout.



- **Enable HAProxy forwarding of requests to Router over TLS**

If you want to:	Encrypt communication between HAProxy and the Gorouter
Then configure the following:	1. Leave **Enable** selected. 2. In the **Certificate Authority for HAProxy Backend** field, specify the Certificate Authority (CA) that signed the certificate you configured in the **Certificate and Private Key for HAProxy and Router** field. **Note:** If you used the **Generate RSA Certificate** link to generate a self-signed certificate, then the CA to specify is the Ops Manager CA, which you can locate at the CA endpoint in the Ops Manager API. 3. Make sure that Gorouter and HAProxy have TLS cipher suites in common in the **TLS Cipher Suites for Router** and **TLS Cipher Suites for HAProxy** fields.
See also:	◦ Terminating SSL/TLS at the Load Balancer and Gorouter ◦ Providing a Certificate for Your SSL/TLS Termination Point ◦ Using the Ops Manager API

- **Disable HAProxy forwarding of requests to Router over TLS**

If you want to:	Use non-encrypted communication between HAProxy and Gorouter, or you are not using HAProxy

Then configure the following:	1. Select **Disable**. 2. If you are not using HAProxy, set the number of HAProxy job instances to `0` on the **Resource Config** page. See [Disable Unused Resources]().
See also:	◦ Terminating SSL/TLS at the Gorouter Only ◦ Terminating SSL/TLS at the Load Balancer Only

11. If you are not using SSL encryption or if you are using self-signed certificates, select **Disable SSL certificate verification for this environment**. Selecting this checkbox also disables SSL verification for route services.

> **Note**: For production deployments, Pivotal does not recommend disabling SSL certificate verification.

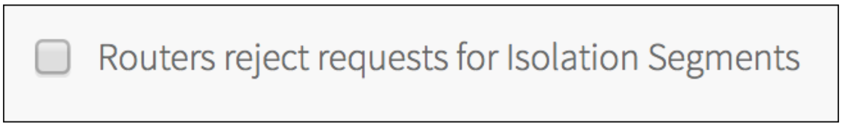
12. (Optional) If you want HAProxy or the Gorouter to reject any HTTP (non-encrypted) traffic, select the **Disable HTTP on HAProxy and Gorouter** checkbox. When selected, HAProxy and Gorouter will not listen on port 80.

13. (Optional) Select the **Disable insecure cookies on the Router** checkbox to set the secure flag for cookies generated by the router.

14. (Optional) To disable the addition of Zipkin tracing headers on the Gorouter, deselect the **Enable Zipkin tracing headers on the router** checkbox. Zipkin tracing headers are enabled by default. For more information about using Zipkin trace logging headers, see Zipkin Tracing in HTTP Headers.

15. (Optional) To stop the Router from writing access logs to local disk, deselect the **Enable Router to write access logs locally** checkbox. You should consider disabling this checkbox for high traffic deployments since logs may not be rotated fast enough and can fill up the disk.

16. By default, the PAS routers handle traffic for applications deployed to an isolation segment created by the PCF Isolation Segment tile. To configure the PAS routers to reject requests for applications within isolation segments, select the **Routers reject requests for Isolation Segments** checkbox.



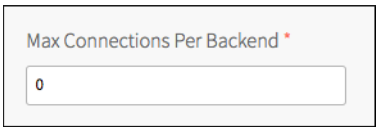
Do not enable this option without deploying routers for each isolation segment. See the following topics for more information:
    - Installing PCF Isolation Segment
    - Sharding Routers for Isolation Segments

17. In the **Choose whether or not to enable route services** section, choose either **Enable route services** or **Disable route services**. Route services are a class of marketplace services that perform filtering or content transformation on application requests and responses. See the Route Services topic for details.

18. (Optional) If you want to limit the number of app connections to the backend, enter a value in the **Max Connections Per Backend** field. You can use this field to prevent a poorly behaving app from all the connections and impacting other apps.

    To choose a value for this field, review the peak concurrent connections received by instances of the most popular apps in your deployment. You can determine the number of concurrent connections for an app from the `httpStartStop` event metrics emitted for each app request.

    If your deployment uses PCF Metrics, you can also obtain this peak concurrent connection information from Network Metrics. The default value is



`500`.

19. Enter a value for **Router Max Idle Keepalive Connections** See Considerations for Configuring max_idle_connections.

20. (Optional) To accommodate larger uploads over connections with high latency, increase the number of seconds in the **Router Timeout to Backends** field.

21. (Optional) Use the **Frontend Idle Timeout for Gorouter and HAProxy** field to help prevent connections from your load balancer to Gorouter or

HAProxy from being closed prematurely. The value you enter sets the duration, in seconds, that Gorouter or HAProxy maintains an idle open connection from a load balancer that supports keep-alive.

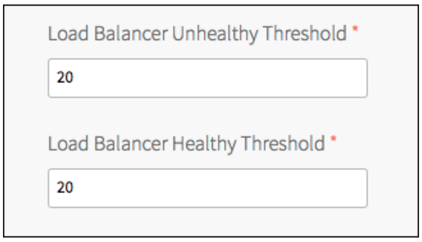
In general, set the value higher than your load balancer's backend idle timeout to avoid the race condition where the load balancer sends a request before it discovers that Gorouter or HAProxy has closed the connection.

See the following table for specific guidance and exceptions to this rule:

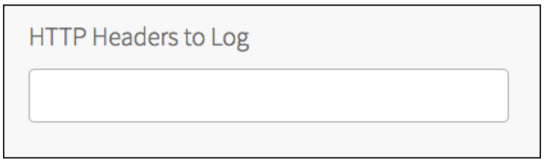
IaaS	Guidance
AWS	AWS ELB has a default timeout of 60 seconds, so Pivotal recommends a value greater than `60`.
Azure	By default, Azure load balancer times out at 240 seconds without sending a TCP RST to clients, so as an exception, Pivotal recommends a value lower than `240` to force the load balancer to send the TCP RST.
GCP	GCP has a default timeout of 600 seconds, so Pivotal recommends a value greater than `600`.
Other	Set the timeout value to be greater than that of the load balancer's backend idle timeout.

22. (Optional) Increase the value of **Load Balancer Unhealthy Threshold** to specify the amount of time, in seconds, that the router continues to accept connections before shutting down. During this period, healthchecks may report the router as unhealthy, which causes load balancers to failover to other routers. Set this value to an amount greater than or equal to the maximum time it takes your load balancer to consider a router instance unhealthy, given contiguous failed healthchecks.

23. (Optional) Modify the value of **Load Balancer Healthy Threshold**. This field specifies the amount of time, in seconds, to wait until declaring the Router instance started. This allows an external load balancer time to register the Router instance as healthy.

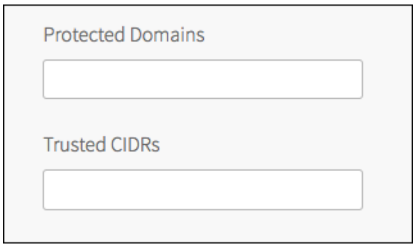


24. (Optional) If app developers in your organization want certain HTTP headers to appear in their app logs with information from the Gorouter, specify them in the **HTTP Headers to Log** field. For example, to support app developers that deploy Spring apps to PCF, you can enter Spring-specific HTTP headers.



25. If you expect requests larger than the default maximum of 16 Kbytes, enter a new value (in bytes) for **HAProxy Request Max Buffer Size**. You may need to do this, for example, to support apps that embed a large cookie or query string values in headers.

26. If your PCF deployment uses HAProxy and you want it to receive traffic only from specific sources, use the following fields:
    - **Protected Domains:** Enter a comma-separated list of domains from which PCF can receive traffic.
    - **Trusted CIDRs:** Optionally, enter a space-separated list of CIDRs to limit which IP addresses from the **Protected Domains** can send traffic to PCF.



27. For Loggregator Port, you must enter `4443` In AWS deployments, port 4443 forwards SSL traffic that supports WebSockets from the ELB. Do not use the default port of `443`

28. For **Container Network Plugin Interface**, ensure **Silk** is selected and review the following fields:

> **Note**: The **External** option exists to support NSX-T integration for vSphere deployments.

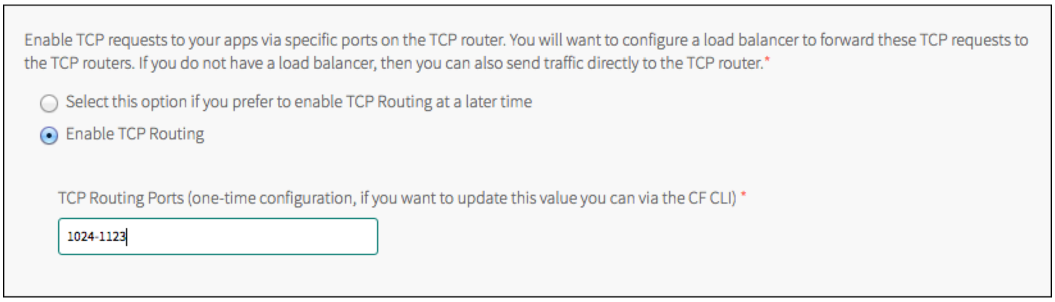
a. (Optional) You can change the value in the **Applications Network Maximum Transmission Unit (MTU)** field. Pivotal recommends setting the MTU value for your application network to `1454`. Some configurations, such as networks that use GRE tunnels, may require a smaller MTU value.
b. (Optional) Enter an IP range for the overlay network in the **Overlay Subnet** box. If you do not set a custom range, Ops Manager uses `10.255.0.0/16`.

> **WARNING**: The overlay network IP range must not conflict with any other IP addresses in your network.

c. Enter a UDP port number in the **VXLAN Tunnel Endpoint Port** box. If you do not set a custom port, Ops Manager uses 4789.
d. For **Denied logging interval**, set the per-second rate limit for packets blocked by either a container-specific networking policy or by Application Security Group rules applied across the space, org, or deployment. This field defaults to `1`.
e. For **UDP logging interval**, set the per-second rate limit for UDP packets sent and received. This field defaults to `100`.
f. To enable logging for app traffic, select **Log traffic for all accepted/denied application packets**. See Manage Logging for Container-to-Container Networking for more information.

29. TCP Routing is disabled by default. To enable this feature, perform the following steps:

a. Select **Enable TCP Routing**
b. In **TCP Routing Ports** enter a range of ports to be allocated for TCP Routes. If you configured AWS for PCF manually, enter `1024-1123` which corresponds to the rules you created for `pcf-tcp-elb` Configuration of this field is only applied on the first deploy, and update updates to the port range are made using the cf CLI. For details about modifying the port range, see the Router Groups topic.



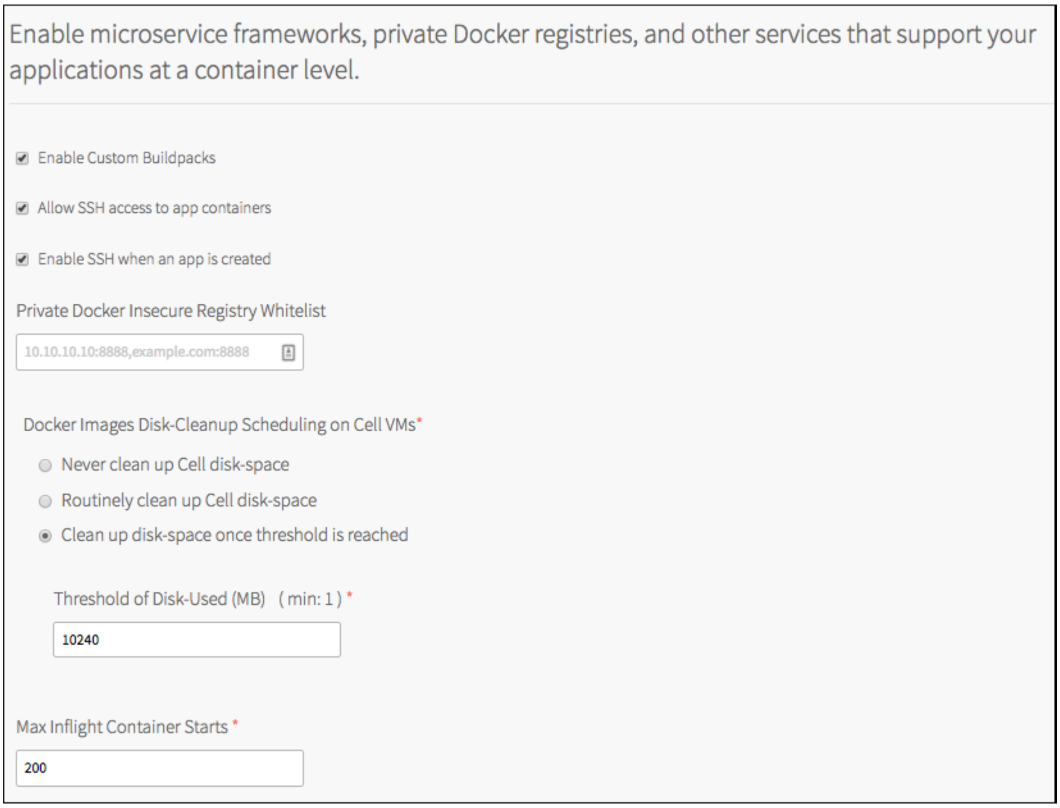
c. For AWS, you also need to specify the name of a TCP ELB in the **LOAD BALANCER** column of TCP Router job of the `Resource Config` screen. You configure this later on in PAS. For more information, see the Configure Router to Elastic Load Balancer topic.

30. (Optional) To disable TCP routing, click **Select this option if you prefer to enable TCP Routing at a later time** For more information, see the Configuring TCP Routing in PAS topic.

31. Click **Save**

Step 5: Configure Application Containers

1. Select **Application Containers**.



2. The **Enable Custom Buildpacks** checkbox governs the ability to pass a custom buildpack URL to the `-b` option of the `cf push` command. By default, this ability is enabled, letting developers use custom buildpacks when deploying apps. Disable this option by disabling the checkbox. For more information about custom buildpacks, refer to the buildpacks section of the PCF documentation.

3. The **Allow SSH access to app containers** checkbox controls SSH access to application instances. Enable the checkbox to permit SSH access across your deployment, and disable it to prevent all SSH access. See the Application SSH Overview topic for information about SSH access permissions at the space and app scope.

4. If you want enable SSH access for new apps by default in spaces that allow SSH, select **Enable SSH when an app is created**. If you deselect the checkbox, developers can still enable SSH after pushing their apps by running `cf enable-ssh APP-NAME`.

5. You can configure Pivotal Application Service (PAS) to run app instances in Docker containers by supplying their IP address range(s) in the **Private Docker Insecure Registry Whitelist** textbox. See the Using Docker Registries topic for more information.

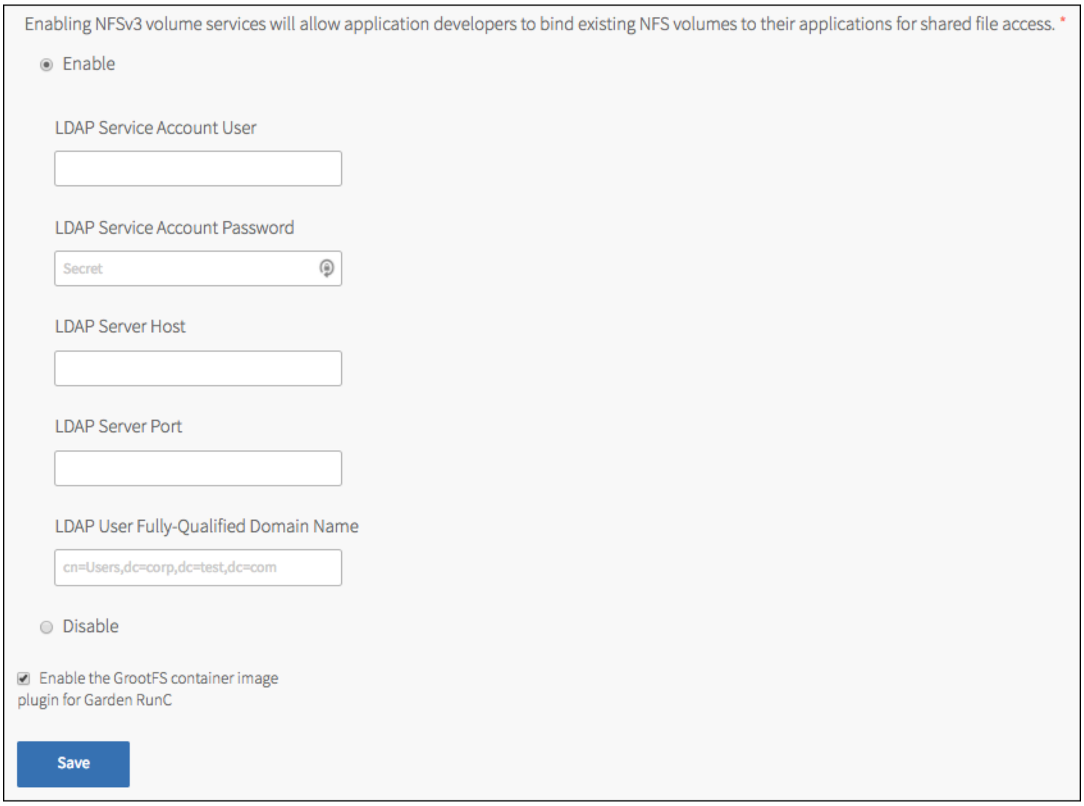
6. Select your preference for **Docker Images Disk-Cleanup Scheduling on Cell VMs**. If you choose **Clean up disk-space once threshold is reached**, enter a **Threshold of Disk-Used** in megabytes. For more information about the configuration options and how to configure a threshold, see Configuring Docker Images Disk-Cleanup Scheduling.

7. Enter a number in the **Max Inflight Container Starts** textbox. This number configures the maximum number of started instances across your deployment's Diego Cells. For more information about this feature, see Setting a Maximum Number of Started Containers.

8. Under **Enabling NFSv3 volume services**, select **Enable** or **Disable**. NFS volume services allow application developers to bind existing NFS volumes to their applications for shared file access. For more information, see the Enabling NFS Volume Services topic.

   > **Note**: In a clean install, NFSv3 volume services is enabled by default. In an upgrade, NFSv3 volume services is set to the same setting as it was in the previous deployment.

9. (Optional) To configure LDAP for NFSv3 volume services, perform the following steps:



- For **LDAP Service Account User**, enter the username of the service account in LDAP that will manage volume services.
- For **LDAP Service Account Password**, enter the password for the service account.
- For **LDAP Server Host**, enter the hostname or IP address of the LDAP server.
- For **LDAP Server Port**, enter the LDAP server port number. If you do not specify a port number, Ops Manager uses 389.
- For **LDAP Server Protocol**, enter the server protocol. If you do not specify a protocol, Ops Manager uses TCP.
- For **LDAP User Fully-Qualified Domain Name**, enter the fully qualified path to the LDAP service account. For example, if you have a service account named `volume-services` that belongs to organizational units (OU) named `service-accounts` and `my-company`, and your domain is named `domain`, the fully qualified path looks like the following:

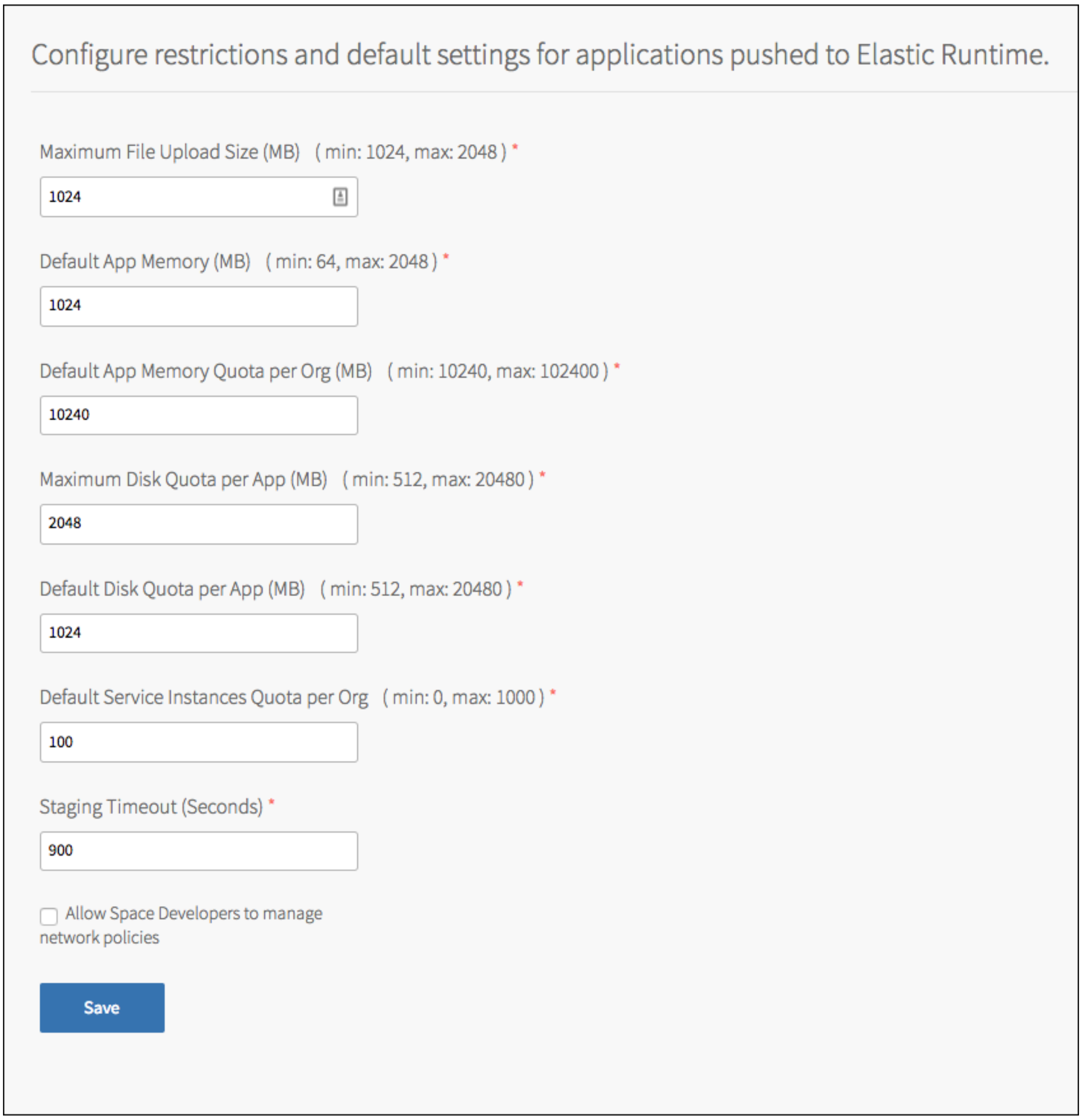
```
CN=volume-services,OU=service-accounts,OU=my-company,DC=domain,DC=com
```

10. By default, PAS manages container images using the GrootFS plugin for Garden-runC. If you experience issues with GrootFS, you can disable the plugin and use the image plugin built into Garden-runC.

11. Click **Save**.

Step 6: Configure Application Developer Controls

1. Select **Application Developer Controls**.



2. Enter the **Maximum File Upload Size (MB)**. This is the maximum size of an application upload.

3. Enter the **Default App Memory (MB)**. This is the amount of RAM allocated by default to a newly pushed application if no value is specified with the cf CLI.

4. Enter the **Default App Memory Quota per Org**. This is the default memory limit for all applications in an org. The specified limit only applies to the first installation of PAS. After the initial installation, operators can use the cf CLI to change the default value.

5. Enter the **Maximum Disk Quota per App (MB)**. This is the maximum amount of disk allowed per application.

> **Note**: If you allow developers to push large applications, PAS may have trouble placing them on Cells. Additionally, in the event of a system upgrade or an outage that causes a rolling deploy, larger applications may not successfully re-deploy if there is insufficient disk capacity. Monitor your deployment to ensure your Cells have sufficient disk to run your applications.

6. Enter the **Default Disk Quota per App (MB)**. This is the amount of disk allocated by default to a newly pushed application if no value is specified with the cf CLI.

7. Enter the **Default Service Instances Quota per Org**. The specified limit only applies to the first installation of PAS. After the initial installation, operators can use the cf CLI to change the default value .

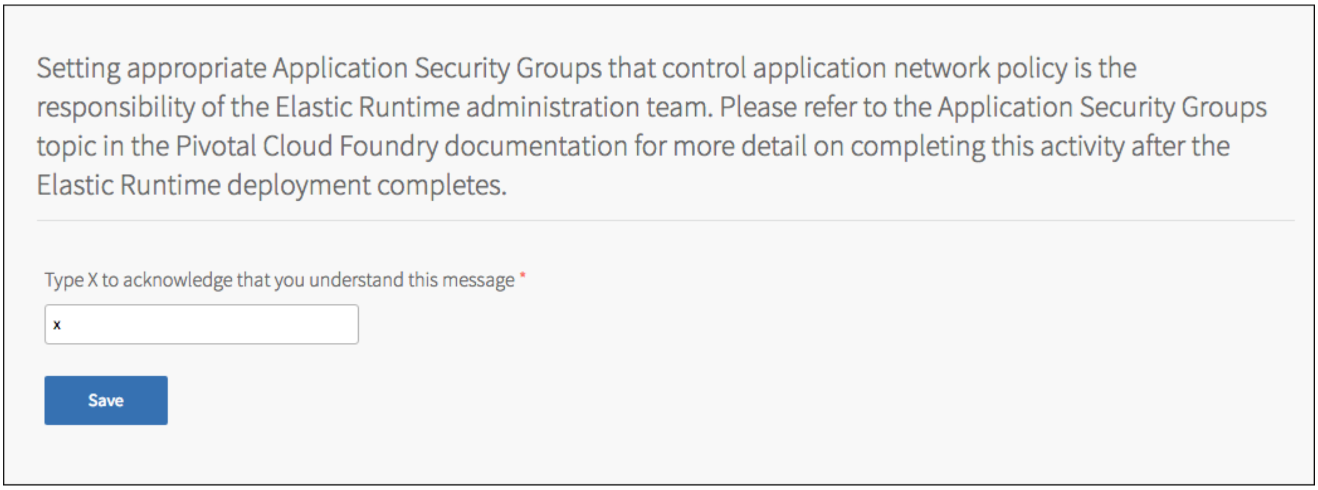
8. Enter the **Staging Timeout (Seconds)**. When you stage an application droplet with the Cloud Controller, the server times out after the number of seconds you specify in this field.

9. Select the **Allow Space Developers to manage network policies** checkbox to permit developers to manage their own network policies for their applications.

10. Click **Save**.

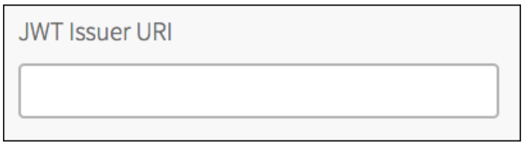
Step 7: Review Application Security Groups

Setting appropriate Application Security Groups is critical for a secure deployment. Type `X` in the box to acknowledge that once the Pivotal Application Service (PAS) deployment completes, you will review and set the appropriate application security groups. See Restricting App Access to Internal PCF Components for instructions.



Step 8: Configure UAA

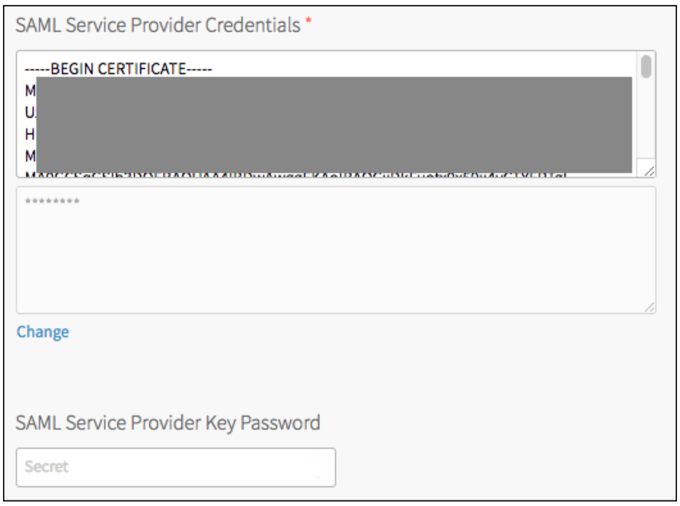
1. Select **UAA**.
2. (Optional) Under **JWT Issuer URI**, enter the URI that UAA uses as the issuer when generating tokens.



3. Under **SAML Service Provider Credentials**, enter a certificate and private key to be used by UAA as a SAML Service Provider for signing outgoing SAML authentication requests. You can provide an existing certificate and private key from your trusted Certificate Authority or generate a self-signed certificate. The following domain must be associated with the certificate: `*.login.YOUR-SYSTEM-DOMAIN`.

> **Note**: The Pivotal Single Sign-On Service and Pivotal Spring Cloud Services tiles require the `*.login.YOUR-SYSTEM-DOMAIN`.

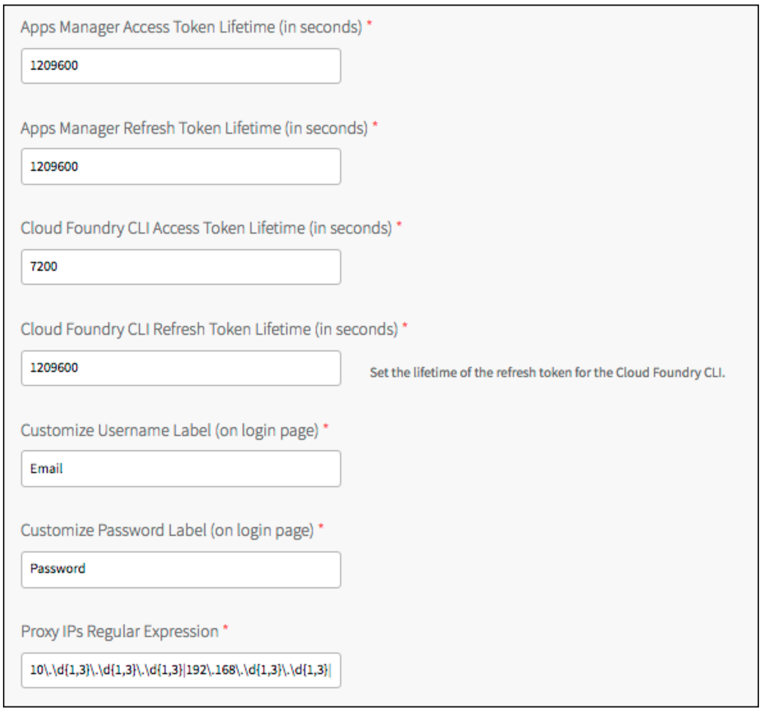
4. If the private key specified under **Service Provider Credentials** is password-protected, enter the password under **SAML Service Provider Key**



Password.

5. For **Signature Algorithm**, choose an algorithm from the dropdown menu to use for signed requests and assertions. The default value is `SHA256`.

6. (Optional) In the **Apps Manager Access Token Lifetime**, **Apps Manager Refresh Token Lifetime**, **Cloud Foundry CLI Access Token Lifetime**, and **Cloud Foundry CLI Refresh Token Lifetime** fields, change the lifetimes of tokens granted for Apps Manager and Cloud Foundry Command Line Interface (cf CLI) login access and refresh. Most deployments use the defaults.



7. (Optional) Customize the text prompts used for username and password from the cf CLI and Apps Manager login popup by entering values for **Customize Username Label (on login page)** and **Customize Password Label (on login page)**.

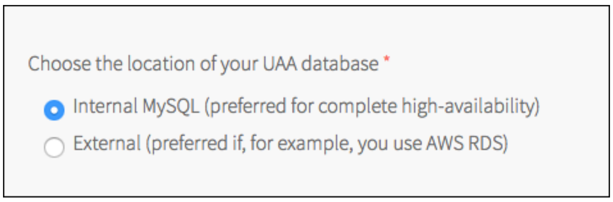
8. (Optional) The **Proxy IPs Regular Expression** field contains a pipe-delimited set of regular expressions that UAA considers to be reverse proxy IP addresses. UAA respects the `x-forwarded-for` and `x-forwarded-proto` headers coming from IP addresses that match these regular expressions. To configure UAA to respond properly to Gorouter or HAProxy requests coming from a public IP address, append a regular expression or regular expressions to match the public IP address.

9. You can configure UAA to use the internal MySQL database provided with PCF, or you can configure an external database provider. Follow the procedures in either the Internal Database Configuration or the External Database Configuration section below.

> **Note**: If you are performing an upgrade, do not modify your existing internal database configuration or you may lose data. You must migrate your existing data before changing the configuration. See Upgrading Pivotal Cloud Foundry for additional upgrade information, and contact Pivotal Support for help.

Internal Database Configuration

1. Select **Internal MySQL**.



1. Click **Save**.

2. Ensure that you complete the “Configure Internal MySQL” step later in this topic to configure high availability and automatic backups for your internal MySQL databases.

External Database Configuration

Note: The exact procedure to create databases depends upon the database provider you select for your deployment. The following procedure uses AWS RDS as an example, but UAA also supports Azure SQL Server.

Warning: Protect whichever database you use in your deployment with a password.

To create your UAA database, perform the following steps:

1. Add the `ubuntu` account key pair from your IaaS deployment to your local SSH profile so you can access the Ops Manager VM. This is the value of the `ops_manager_ssh_private_key` from the Terraform output.

```
$ ssh-add aws-keypair.pem
```

2. SSH in to your Ops Manager using the Ops Manager FQDN and the username `ubuntu`:

```
$ ssh ubuntu@OPS-MANAGER-FQDN
```

3. Log in to your MySQL database instance using the appropriate hostname and user login values configured in your IaaS account. For example, to log in to your AWS RDS instance, run the following MySQL command:

```
$ mysql --host=RDSHOSTNAME --user=RDSUSERNAME --password=RDSPASSWORD
```

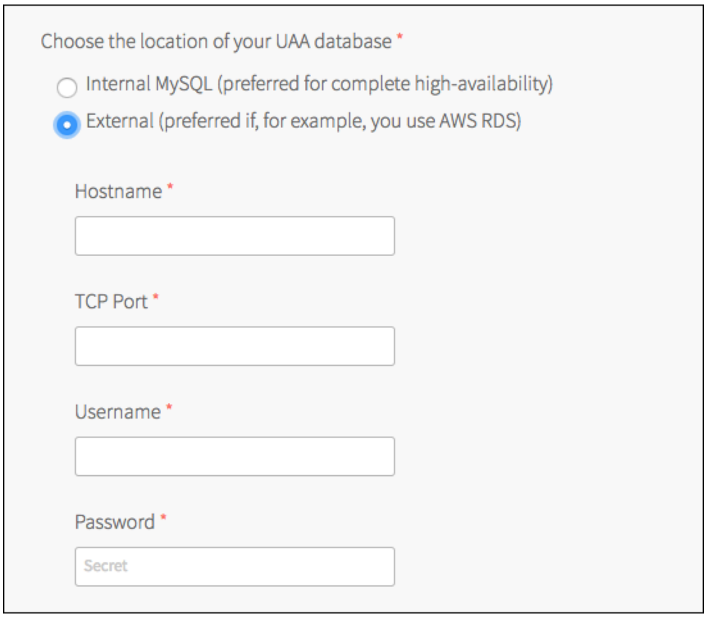
4. Run the following MySQL commands to create a database for UAA:

```
CREATE database uaa;
```

5. Type `exit` to quit the MySQL client, and `exit` again to close your connection to the Ops Manager VM.

6. Reboot the RDS instance in the AWS console.

7. From the **UAA** section in Pivotal Application Service (PAS), select **External**.



8. For **Hostname**, enter the hostname of the database server. This is the value from the `rds_address` key in the Terraform output.

9. For **TCP Port**, enter the port of the database server. This is the value from the `rds_port` key in the Terraform output.

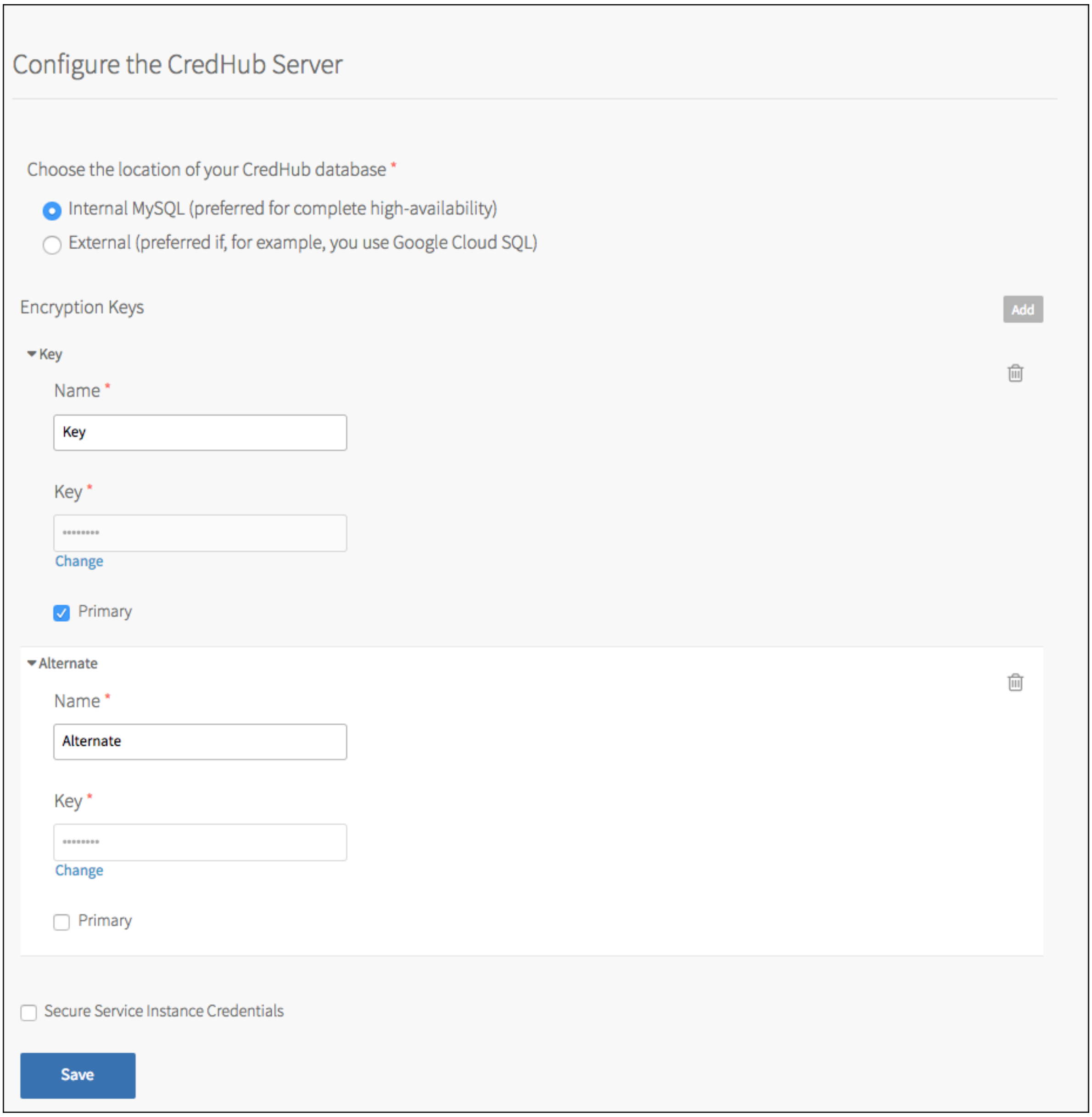
10. For **User Account and Authentication database username**, specify the username that can access this database on the database server. This is the value from the `rds_username` key in the Terraform output.

11. For **User Account and Authentication database password**, specify a password for the provided username. This is the value from the `rds_password` key in the Terraform output.

12. Click **Save**.

Step 9: (Optional) Configure CredHub

1. Select **Credhub**.



2. Choose the location of your CredHub Database. PAS includes this CredHub database for services to store their service instance credentials.

   a. If you chose **External**, enter the following:
      - **Hostname**: The IP address of your database server. This is the value from the `PcfRdsAddress` key in the AWS output.
      - **TCP Port**: The port of your database server. This is the value from the `PcfRdsPort` key in the AWS output..
      - **Username**: The value from the `PcfRdsUsername` key in the AWS output.
      - **Password**: The value from the `PcfRdsPassword` key in the AWS output.
      - **Database CA Certificate**: Enter a certificate to use for encrypting traffic to and from the database.

3. Under **Encryption Keys**, specify a key to use for encrypting and decrypting the values stored in the CredHub database.
   - **Name**: Enter the name of the key.
   - **Key**: Enter a key that is at least 20 characters in length.
   - **Primary**: Select this checkbox to use this key as your primary key.

> **Note**: Ensure that you only mark one key as **Primary**. The UI includes an **Add** button to add more keys to support key rotation. For more information, see the Rotating PAS CredHub Encryption Keys topic.

4. If your deployment uses any PCF services that support storing service instance credentials in CredHub and you want to enable this feature, select the **Secure Service Instance Credentials** checkbox.

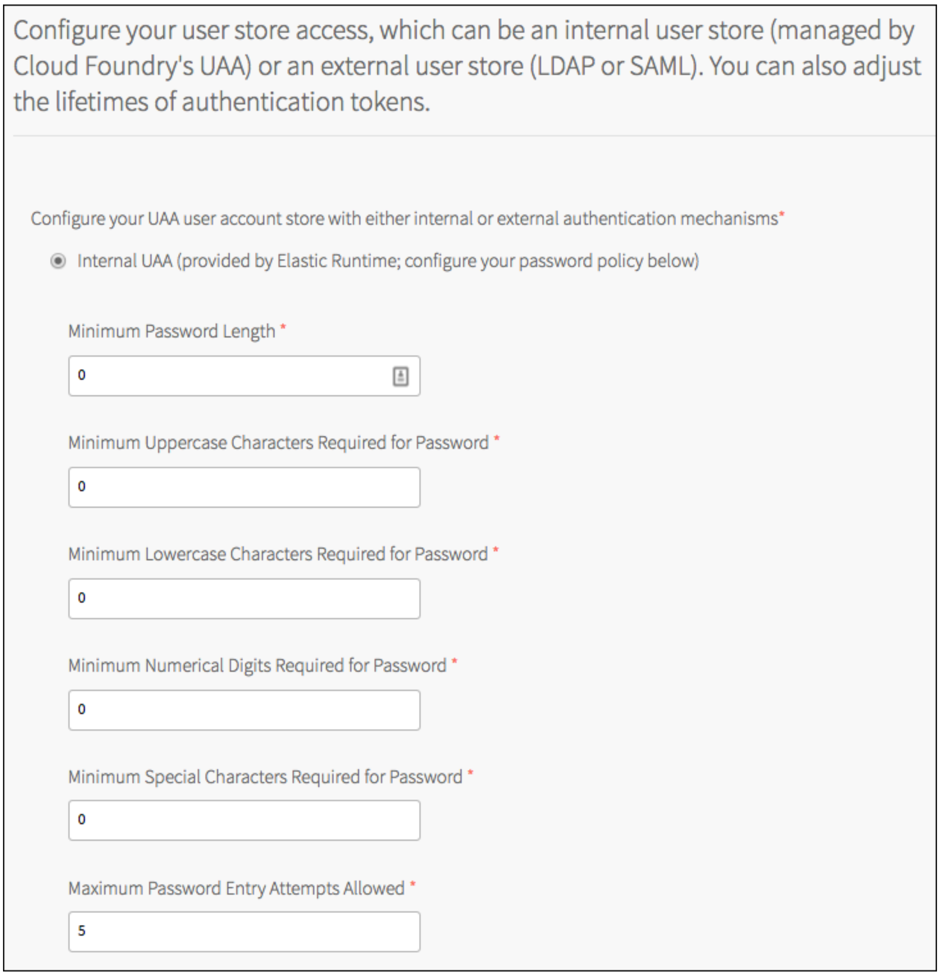
5. Select the **Resource Config** pane.

6. Under the **Job** column of the **CredHub** row, set the number of instances to `2`. This is the minimum instance count required for high availability.

> **Note**: To use the Runtime CredHub feature, you must follow the additional steps in Securing Service Instance Credentials with Runtime CredHub.

Step 10: Configure Authentication and Enterprise SSO

1. Select **Authentication and Enterprise SSO**.



2. To authenticate user sign-ons, your deployment can use one of three types of user database: the UAA server's internal user store, an external SAML identity provider, or an external LDAP server.
   - To use the internal UAA, select the **Internal** option and follow the instructions in the Configuring UAA Password Policy topic to configure your password policy.
   - To connect to an external identity provider through SAML, scroll down to select the **SAML Identity Provider** option and follow the instructions in the Configuring PCF for SAML section of the *Configuring Authentication and Enterprise SSO for Pivotal Application Service (PAS)* topic.
   - To connect to an external LDAP server, scroll down to select the **LDAP Server** option and follow the instructions in the Configuring LDAP section of the *Configuring Authentication and Enterprise SSO for PAS* topic.

3. Click **Save**.

Step 11: Configure System Databases

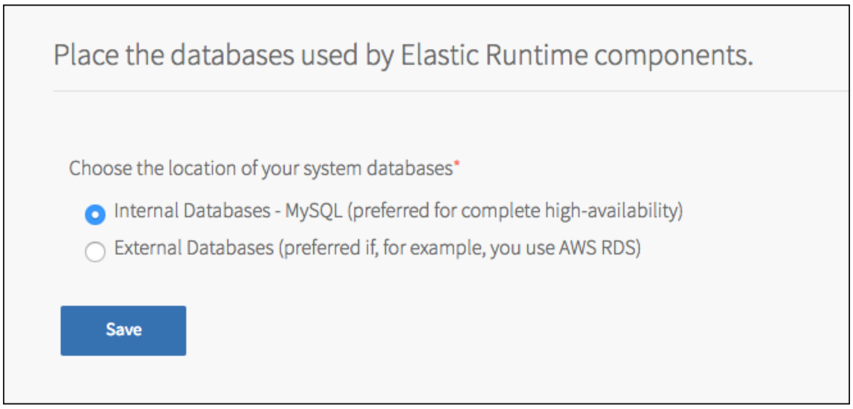
You can configure PAS to use the internal MySQL database provided with PCF, or you can configure an external database provider for the databases required by PAS.

> **Note**: If you are performing an upgrade, do not modify your existing internal database configuration or you may lose data. You must migrate your existing data first before changing the configuration. See Upgrading Pivotal Cloud Foundry for additional upgrade information.

Internal Database Configuration

If you want to use internal databases for your deployment, perform the following steps:

1. Select **Databases**.



2. Select **Internal Databases - MySQL**.
3. Click **Save**.

Then proceed to Step 12: (Optional) Configure Internal MySQL to configure high availability and automatic backups for your internal MySQL databases.

Create External System Databases

If you want to use an external database provider for your PAS databases, you must first create the databases on the RDS instance provided by the Terraform templates.

To create the required databases on an AWS RDS instance, perform the following steps.

1. Add the AWS-provided key pair to your SSH profile so that you can access the Ops Manager VM. This is the value of the `ops_manager_ssh_private_key` from the Terraform output.

```
ssh-add aws-keypair.pem
```

2. SSH in to your Ops Manager using the Ops Manager FQDN and the username `ubuntu`:

```
ssh ubuntu@OPS_MANAGER_FQDN
```

3. Run the following terminal command to log in to your RDS instance through the MySQL client, using values from the terraform output to fill in the following keys:

```
mysql --host=rds_address --user=rds_username --password=rds_password
```

4. Run the following MySQL commands to create databases for the seven PAS components that require a relational database:

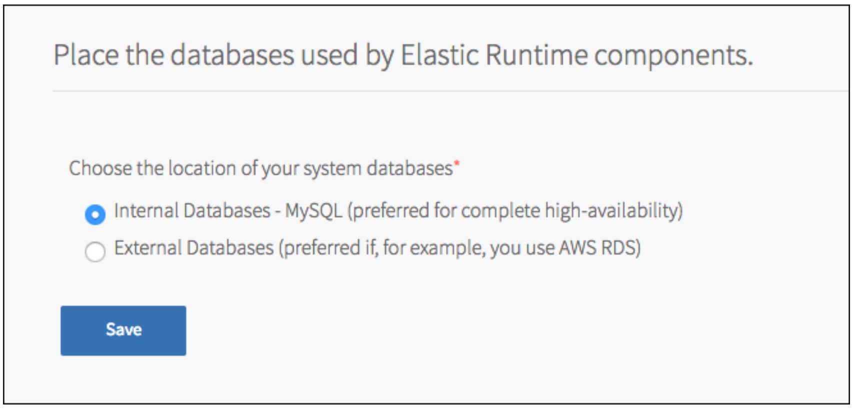
```
CREATE database ccdb;
CREATE database notifications;
CREATE database autoscale;
CREATE database app_usage_service;
CREATE database routing;
CREATE database diego;
CREATE database account;
CREATE database nfsvolume;
CREATE database networkpolicyserver;
CREATE database silk;
CREATE database locket;
CREATE database credhub;
```

5. Type `exit` to quit the MySQL client, and `exit` again to close your connection to the Ops Manager VM.

6. Reboot the RDS instance in the AWS console.

7. In PAS, select **Databases**.

8. Select the **External Databases** option.



9. For the **Hostname** and **TCP Port** fields, complete the following fields:

PAS Field	terraform output
Hostname	`rds_address`
TCP Port	`rds_port`

10. For each **database username** and **database password** field, complete the following fields:

PAS Field	terraform output
DATABASE-NAME database username	`rds_username`
DATABASE-NAME database password	`rds_password`

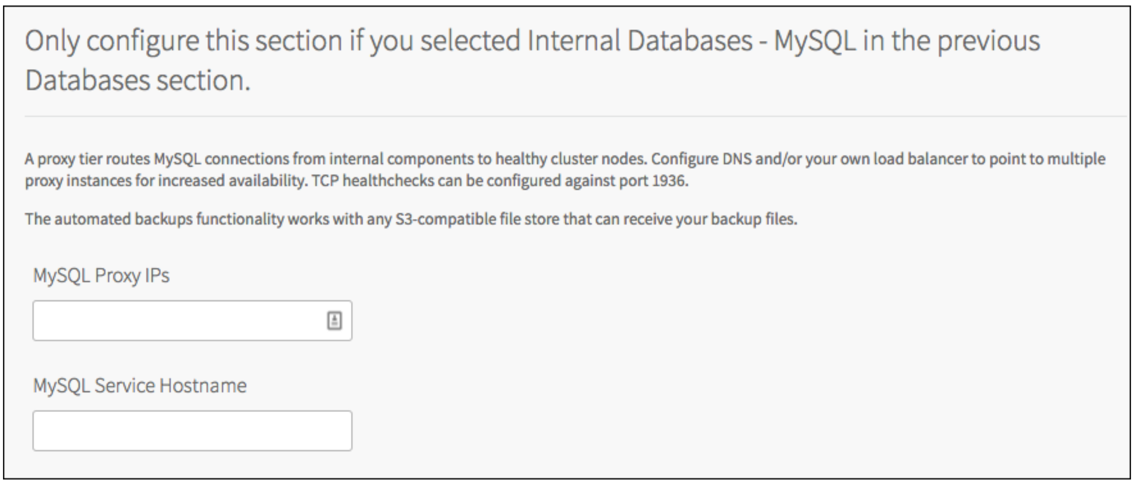
11. Click **Save**.

Step 13: (Optional) Configure Internal MySQL

Note: You only need to configure this section if you have selected **Internal Databases - MySQL** in the **Databases** section.

1. Select **Internal MySQL**.

2. In the **MySQL Proxy IPs** field, enter one or more comma-delimited IP addresses that are not in the reserved CIDR range of your network. If a MySQL node fails, these proxies re-route connections to a healthy node. See the MySQL Proxy topic for more information.



3. For **MySQL Service Hostname**, enter an IP address or hostname for your load balancer. If a MySQL proxy fails, the load balancer re-routes connections to a healthy proxy. If you leave this field blank, components are configured with the IP address of the first proxy instance entered above.

   > **Warning**: You must configure a load balancer to achieve complete high-availability.

4. In the **Replication canary time period** field, leave the default of 30 seconds or modify the value based on the needs of your deployment. Lower numbers cause the canary to run more frequently, which means that the canary reacts more quickly to replication failure but adds load to the database.

5. In the **Replication canary read delay** field, leave the default of 20 seconds or modify the value based on the needs of your deployment. This field configures how long the canary waits, in seconds, before verifying that data is replicating across each MySQL node. Clusters under heavy load can experience a small replication lag as write-sets are committed across the nodes.

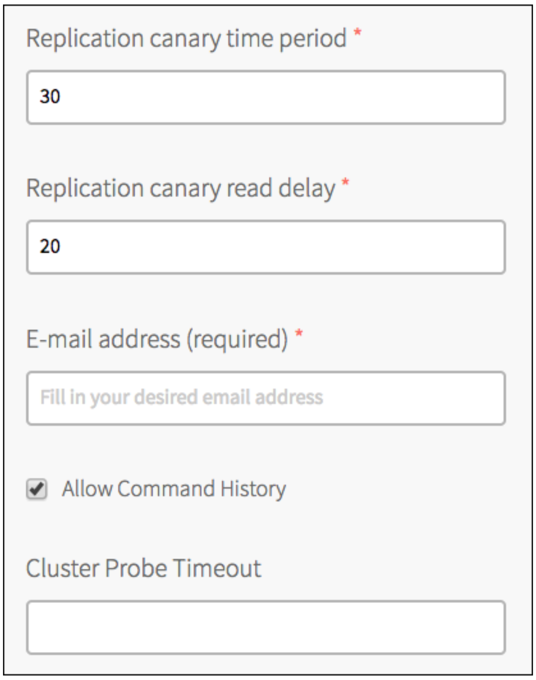
6. (**Required**): In the **E-mail address** field, enter the email address where the MySQL service sends alerts when the cluster experiences a replication issue or when a node is not allowed to auto-rejoin the cluster.

7. To prohibit the creation of command line history files on the MySQL nodes, disable the **Allow Command History** checkbox.

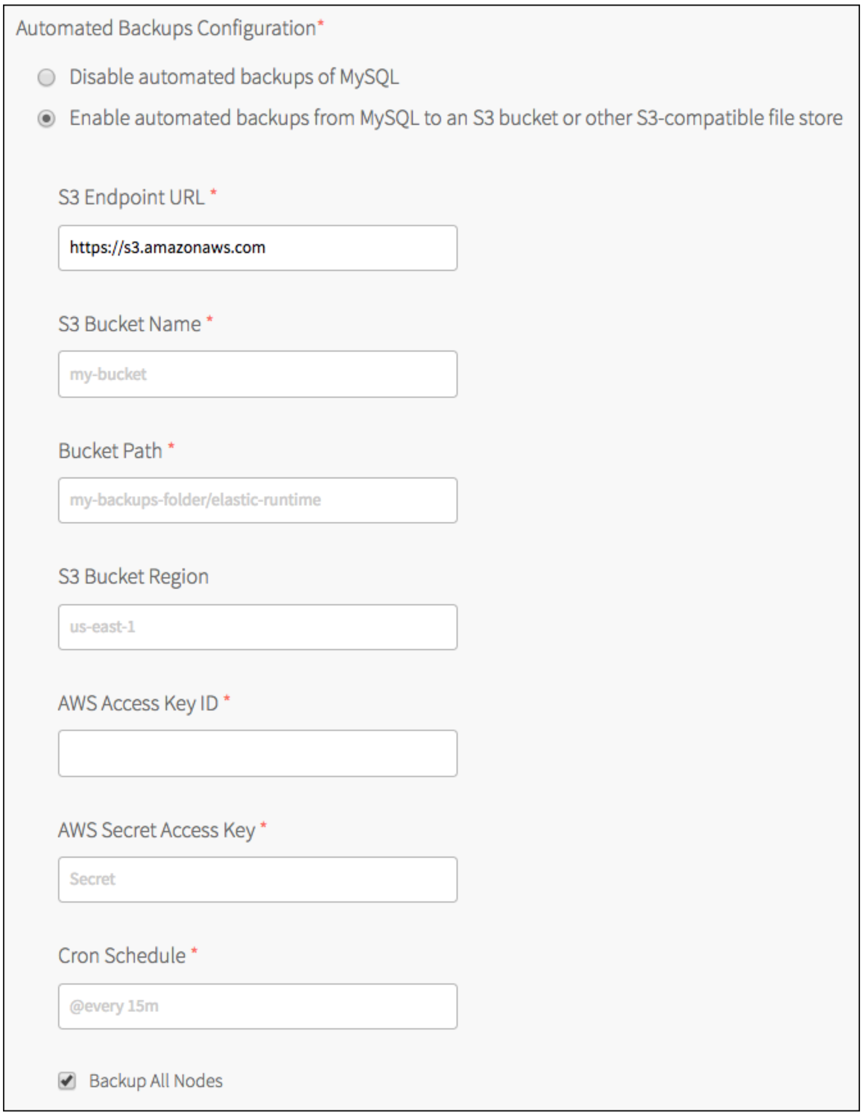
8. To allow the admin and roadmin to connect from any remote host, enable the **Allow Remote Admin Access** checkbox. When the checkbox is disabled, admins must `bosh ssh` into each MySQL VM to connect as the MySQL super user.

   > **Note:** Network configuration and Application Security Groups restrictions may still limit a client's ability to establish a connection with the databases.

9. For **Cluster Probe Timeout**, enter the maximum amount of time, in seconds, that a new node will search for existing cluster nodes. If left blank, the default value is 10 seconds.



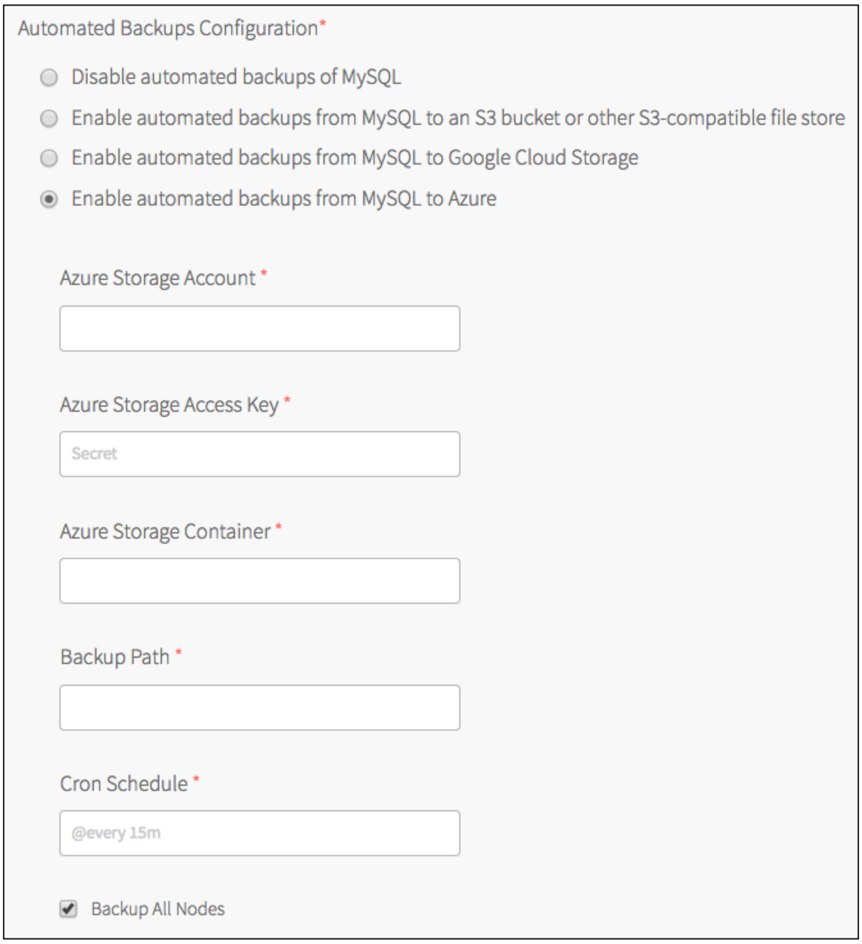
10. Under **Automated Backups Configuration**, choose one of three options for MySQL backups:
    - **Disable automatic backups of MySQL** disables automatic backups, but you can still deploy the Backup Prepare Node if you use BOSH Backup and Restore to back up your MySQL database. For more information, see the Backing Up Pivotal Cloud Foundry with BBR topic.
    - **Enable automated backups from MySQL to an S3 bucket or other S3-compatible file store** saves your backups to an existing Amazon Web Services (AWS) or Ceph S3-compatible blobstore.



This option requires the following fields:

- For **S3 Bucket Name**, enter the name of your S3 bucket. Do not include an `s3://` prefix, a trailing `/`, or underscores. If the bucket does not already exist, it will be created automatically.
- For **Bucket Path**, specify a folder within the bucket to hold your MySQL backups. Do not include a trailing `/`.
- For **S3 Bucket Region**, enter the AWS region where the bucket is located, such as `us-east-1`.
- For **AWS Access Key ID** and **AWS Secret Access Key**, enter your AWS or Ceph credentials.
- For **Cron Schedule**, enter a valid cron expression to schedule your automated backups. Cron uses your computer's local time zone.
- Enable **Backup All Nodes** to make unique backups from each instance of the MySQL server rather than just the first MySQL server instance.

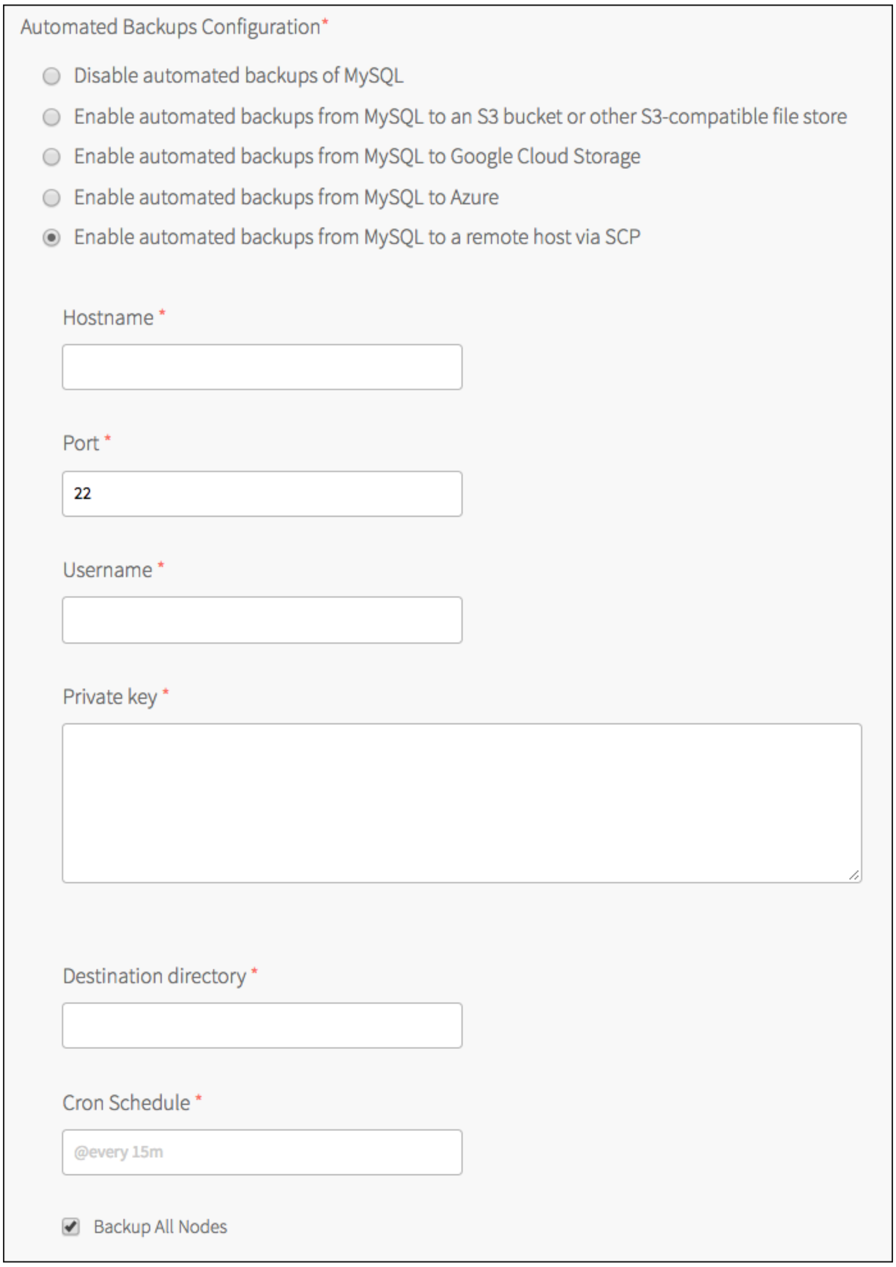
- **Enable automated backups from MySQL to Azure** saves your backups to Azure.



This option requires the following fields:

- For **Azure Storage Account**, enter the name of an existing Azure storage account where backups will be uploaded. For more information about creating and managing an Azure storage account, see the Azure documentation.
- For **Azure Storage Access Key**, enter an Azure storage access key for the storage account.
- For **Azure Storage Container**, enter the name of an existing Azure storage container that will store the backups.
- For **Backup Path**, enter the path within the Azure storage container where backups will be uploaded.
- For **Cron Schedule**, enter a valid cron expression to schedule your automated backups. Cron uses your computer's local time zone.
- Enable **Backup All Nodes** to make unique backups from each instance of the MySQL server rather than just the first MySQL server instance.

- **Enable automated backups from MySQL to a remote host via SCP**saves your backups to a remote host using secure copy protocol (SCP).



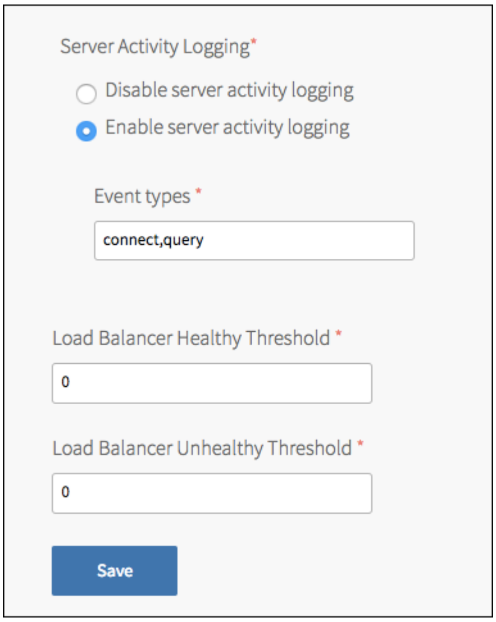
This option requires the following fields:

- For **Hostname**, enter the name of your SCP host.
- For **Port**, enter your SCP port. This should be the TCP port that your SCP host uses for SSH. The default port is `22`.
- For **Username**, enter your SSH username for the SCP host.
- For **Private key**, paste in your SSH private key.
- For **Destination directory**, enter the directory on the SCP host where you want to save backup files.
- For **Cron Schedule**, enter a valid cron expression to schedule your automated backups. Cron uses your computer's local time zone.
- Enable **Backup All Nodes** to make unique backups from each instance of the MySQL server rather than just the first MySQL server instance.

> **Note**: If you choose to enable automated MySQL backups, set the number of instances for the **Backup Prepare Node** under the **Resource Config** section of the Pivotal Application Service (PAS) tile to `1`.

11. If you want to log audit events for internal MySQL, select **Enable server activity logging** under **Server Activity Logging**.

    a. For the **Event types** field, you can enter the events you want the MySQL service to log. By default, this field includes `connect` and `query`, which tracks who connects to the system and what queries are processed. For more information, see the Logging Events section of the MariaDB documentation.



12. Enter values for the following fields:
    - **Load Balancer Healthy Threshold**: Specifies the amount of time, in seconds, to wait until declaring the MySQL proxy instance started. This allows an external load balancer time to register the instance as healthy.
    - **Load Balancer Unhealthy Threshold**: Specifies the amount of time, in seconds, that the MySQL proxy continues to accept connections before shutting down. During this period, the healthcheck reports as unhealthy to cause load balancers to fail over to other proxies. You must enter a value greater than or equal to the maximum time it takes your load balancer to consider a proxy instance unhealthy, given repeated failed healthchecks.

13. If you want to enable the MySQL interruptor feature, select the checkbox to **Prevent node auto re-join**. This feature stops all writes to the MySQL database if it notices an inconsistency in the dataset between the nodes. For more information, see the Interruptor section in the MySQL for PCF documentation.

14. Click **Save**.

Step 14: Configure File Storage

To minimize system downtime, Pivotal recommends using highly resilient and redundant *external* filestores for your Pivotal Application Service (PAS) file storage.

When configuring file storage for the Cloud Controller in PAS, you can select one of the following:

- Internal WebDAV filestore
- External S3-compatible or Ceph-compatible filestore
- External Google Cloud Storage
- External Azure Cloud Storage

For production-level PCF deployments on AWS, Pivotal recommends selecting the **External S3-Compatible File Store**. For more information about production-level PCF deployments on AWS, see the Reference Architecture for Pivotal Cloud Foundry on AWS.

For additional factors to consider when selecting file storage, see the Considerations for Selecting File Storage in Pivotal Cloud Foundry topic.

Internal Filestore

Internal file storage is only appropriate for small, non-production deployments.

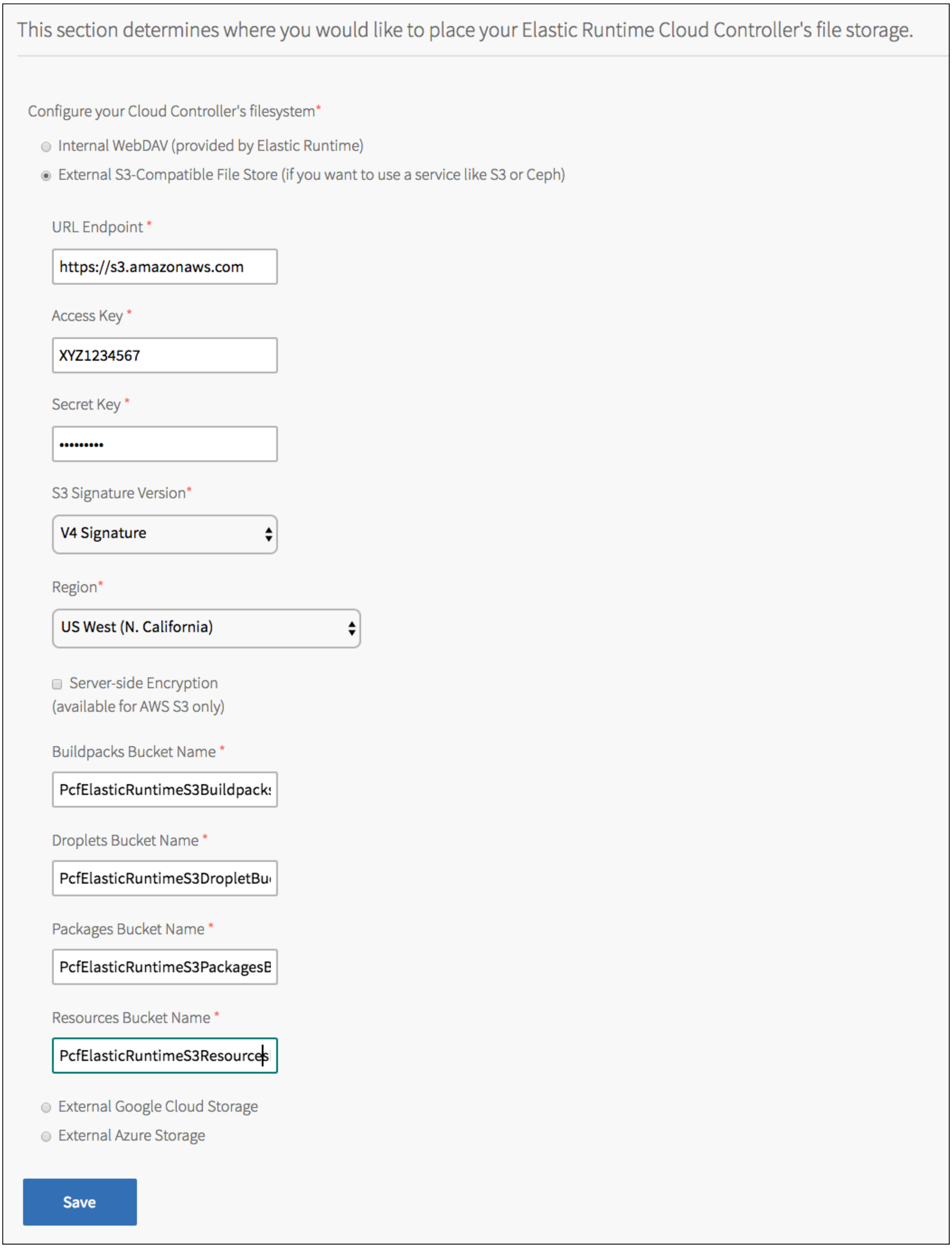
To use the PCF internal filestore, perform the following steps:

1. In the Pivotal Application Service (PAS) tile, select **File Storage**.
2. Select **Internal WebDAV**, and click **Save**.

External S3 or Ceph Filestore

To use an external S3-compatible filestore for your PAS file storage, perform the following steps:

1. In the PAS tile, select **File Storage**.



2. Select the **External S3-Compatible Filestore** option and complete the following fields:
   - For **URL Endpoint**:

1. In a browser, open the Amazon Simple Storage Service (Amazon S3) table.
2. Prepend `https://` to the **Endpoint** for your region and copy it into the Ops Manager **URL Endpoint** field. For example, in the **us-west-2** region, use `https://s3-us-west-2.amazonaws.com/`.

- For **S3 Signature Version** and **Region**, use the **V4 Signature** values. AWS recommends using Signature Version 4.
- Select **Server-side Encryption (available for AWS S3 only)** to encrypt the contents of your S3 filestore. See the AWS S3 documentation for more information.
- Complete the following fields:

Ops Manager Field	terraform output
Buildpacks Bucket Name	`pas_buildpackets_bucket`
Droplets Bucket Name	`pas_droplets_bucket`
Packages Bucket Name	`pas_packages_bucket`
Resources Bucket Name	`pas_resources_bucket`
Access Key ID	`iam_user_access_key`
AWS Secret Key	`iam_user_secret_access_key`

Note: For more information regarding AWS S3 Signatures, see the Authenticating Requests documentation.

3. Click **Save**.

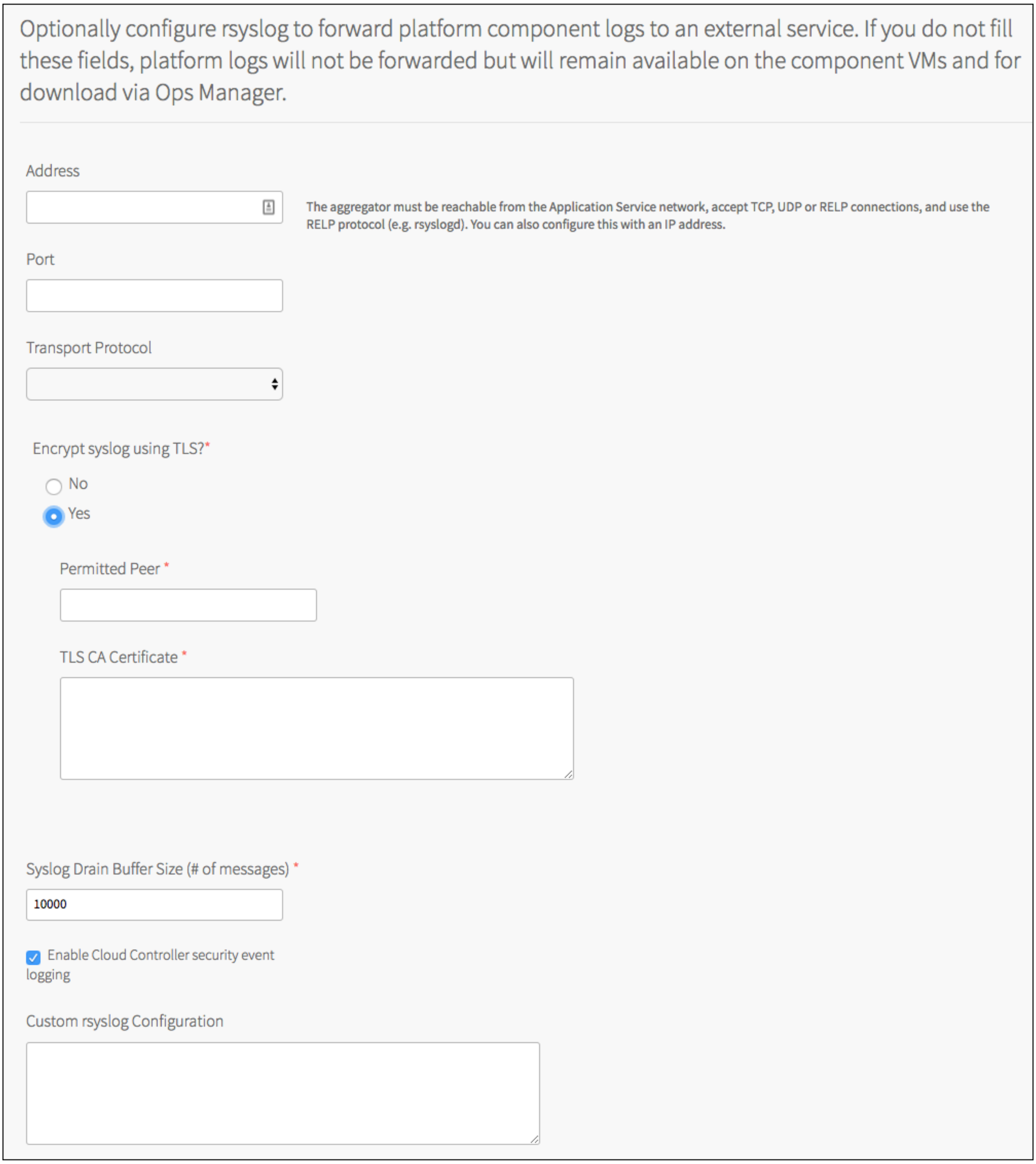
Other IaaS Storage Options

Google Cloud Storage and Azure Storage are also available as file storage options, but are not recommended for typical PCF on AWS installations.

Step 15: (Optional) Configure System Logging

If you forward logging messages to an external Reliable Event Logging Protocol (RELP) server, complete the following steps:

1. Select the **System Logging** section that is located within your PAS **Settings** tab.



2. Enter the IP address of your syslog server in **Address**.

3. Enter the port of your syslog server in **Port**. The default port for a syslog server is `514`.

> **Note**: The host must be reachable from the PAS network, accept TCP connections, and use the RELP protocol. Ensure your syslog server listens on external interfaces.

4. Select a **Transport Protocol** to use when forwarding logs.

5. If you plan to use TLS encryption when sending logs to the remote server, select **Yes** when answering the **Encrypt syslog using TLS?** question.

    a. In the **Permitted Peer** field, enter either the name or SHA1 fingerprint of the remote peer.
    b. In the **TLS CA Certificate** field, enter the TLS CA Certificate for the remote server.

6. For the **Syslog Drain Buffer Size**, enter the number of messages the Doppler server can hold from Metron agents before the server starts to drop them. See the Loggregator Guide for Cloud Foundry Operators topic for more details.

7. If you want to include security events in your log stream, select the **Enable Cloud Controller security event logging** checkbox. This logs all API requests, including the endpoint, user, source IP address, and request result, in the Common Event Format (CEF).

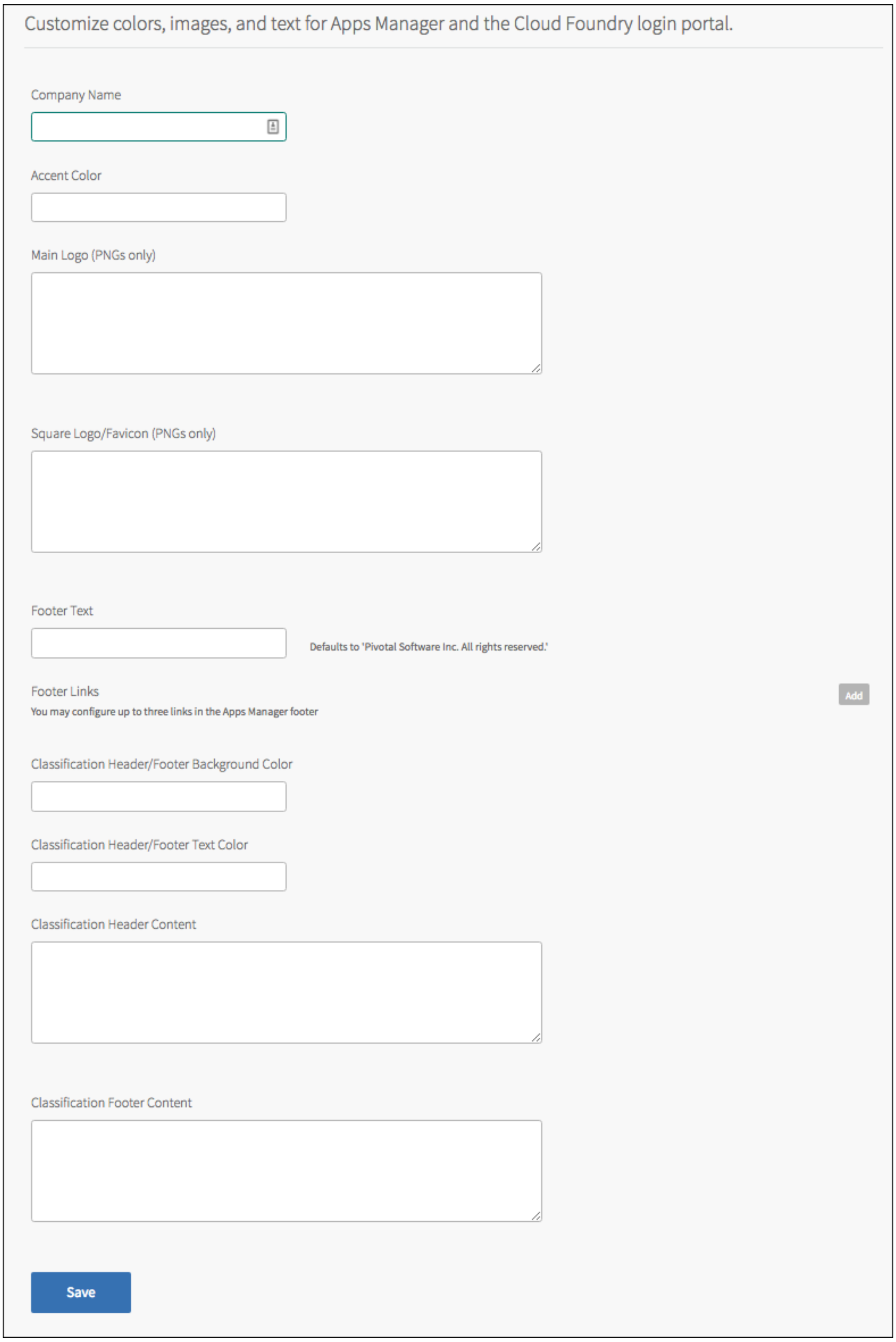
8. If you want to specify a custom syslog formatting rule, enter it in the **Custom rsyslog Configuration** field in RainerScript syntax.

9. Click **Save**.

Step 16: (Optional) Customize Apps

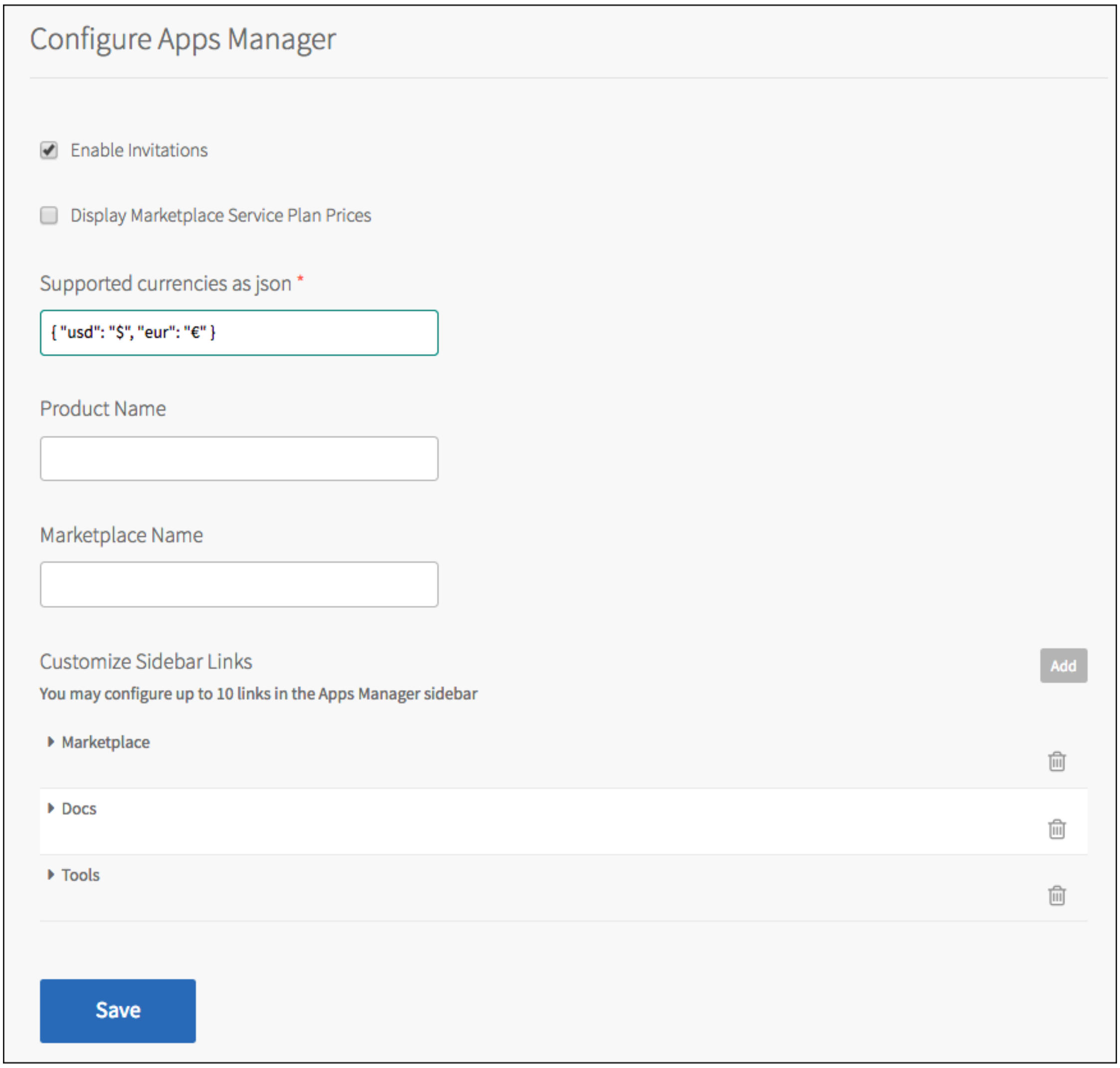
The **Custom Branding** and **Apps Manager** sections customize the appearance and functionality of Apps Manager. Refer to Custom Branding Apps Manager for descriptions of the fields on these pages and for more information about customizing Apps Manager.

1. Select **Custom Branding**. Use this section to configure the text, colors, and images of the interface that developers see when they log in, create an account, reset their password, or use Apps Manager.



2. Click **Save** to save your settings in this section.

3. Select **Apps Manager**.



4. Select **Enable Invitations** to enable invitations in Apps Manager. Space Managers can invite new users for a given space, Org Managers can invite new users for a given org, and Admins can invite new users across all orgs and spaces. See the Inviting New Users section of the *Managing User Roles with Apps Manager* topic for more information.

5. Select **Display Marketplace Service Plan Prices** to display the prices for your services plans in the Marketplace.

6. Enter the **Supported currencies as json** to appear in the Marketplace. Use the format `{"CURRENCY-CODE":"SYMBOL"}`. This defaults to `{"usd": "$", "eur": "€"}`.

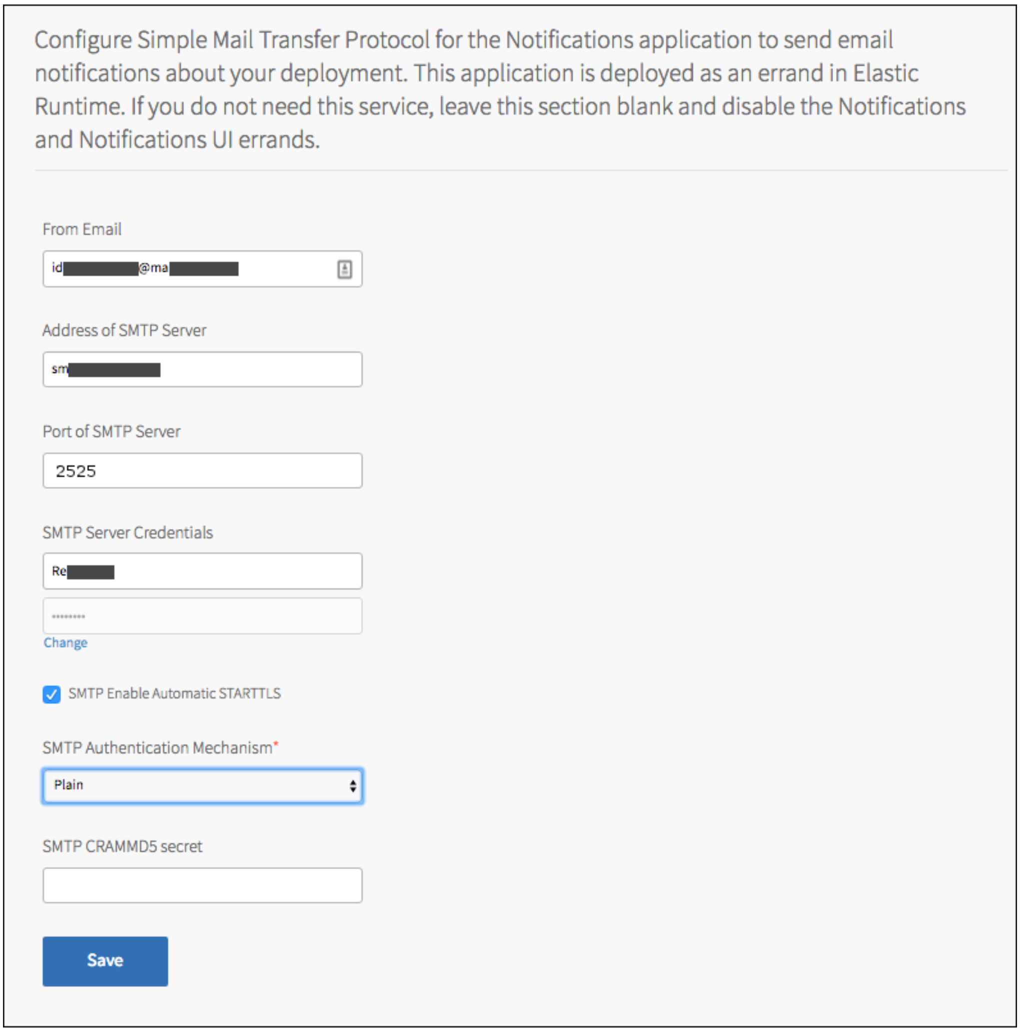
7. Use **Product Name**, **Marketplace Name**, and **Customize Sidebar Links** to configure page names and sidebar links in the **Apps Manager** and **Marketplace** pages.

8. Click **Save** to save your settings in this section.

Step 17: (Optional) Configure Email Notifications

PAS uses SMTP to send invitations and confirmations to Apps Manager users. You must complete the **Email Notifications** page if you want to enable end-user self-registration.

1. Select **Email Notifications**.



2. Enter your reply-to and SMTP email information.

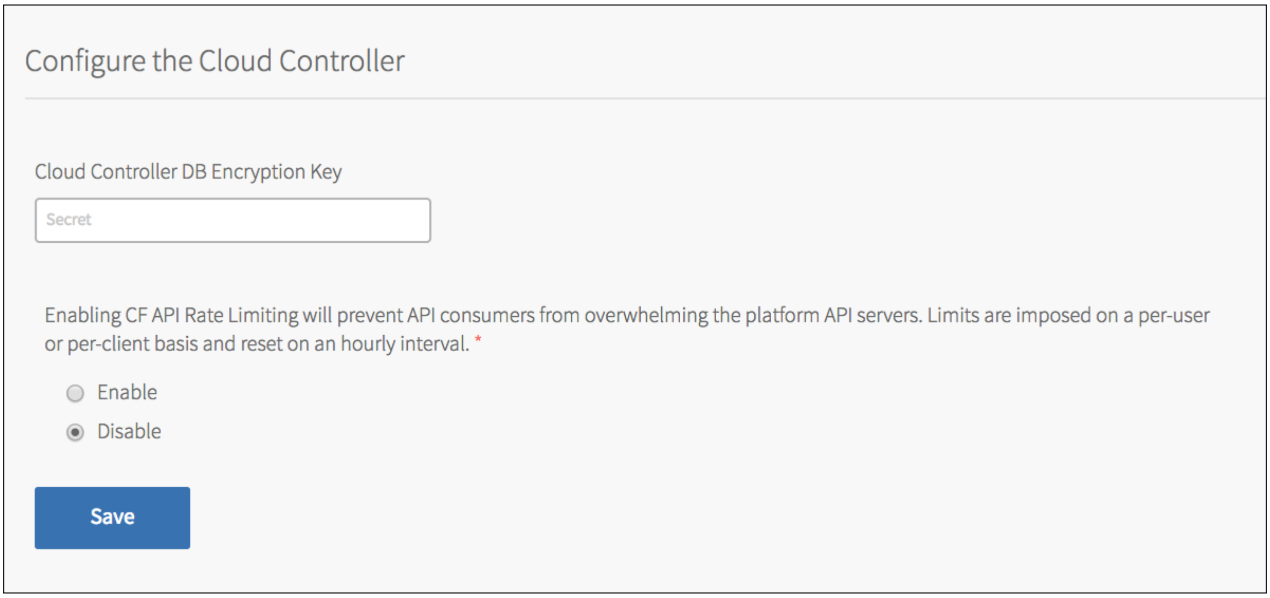
3. For **SMTP Authentication Mechanism**, select `none`.

4. Click **Save**.

> **Note**: If you do not configure the SMTP settings using this form, the administrator must create orgs and users using the cf CLI tool. See Creating and Managing Users with the cf CLI for more information.

Step 18: Configure Cloud Controller

1. Click **Cloud Controller**.



2. Enter your **Cloud Controller DB Encryption Key** if all of the following are true:
   - You deployed Pivotal Application Service (PAS) previously.
   - You then stopped PAS or it crashed.
   - You are re-deploying PAS with a backup of your Cloud Controller database.

   See Backing Up Pivotal Cloud Foundry for more information.

3. CF API Rate Limiting prevents API consumers from overwhelming the platform API servers. Limits are imposed on a per-user or per-client basis and reset on an hourly interval.

   To disable CF API Rate Limiting, select **Disable** under **Enable CF API Rate Limiting**. To enable CF API Rate Limiting, perform the following steps:

   a. Under **Enable CF API Rate Limiting**, select **Enable**.
   b. For **General Limit**, enter the number of requests a user or client is allowed to make over an hour interval for all endpoints that do not have a custom limit. The default value is `2000`.
   c. For **Unauthenticated Limit**, enter the number of requests an unauthenticated client is allowed to make over an hour interval. The default value is `100`.

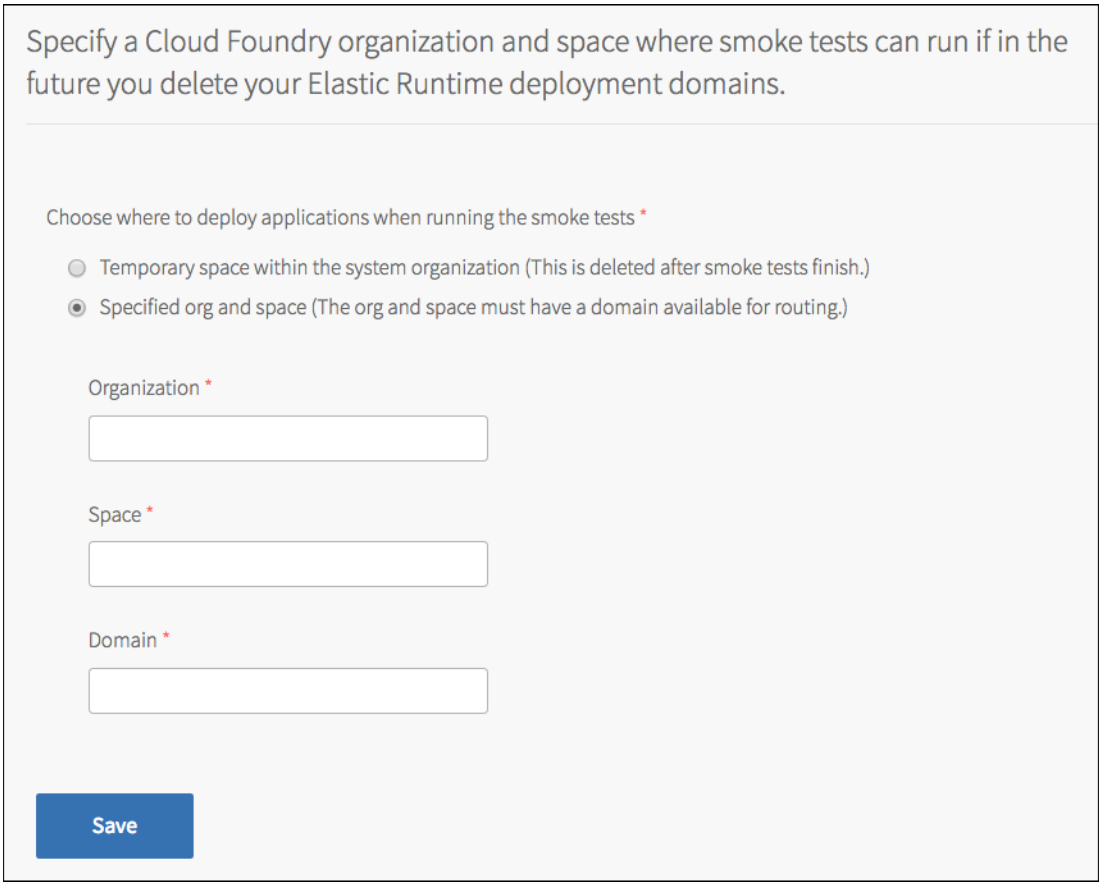
4. Click **Save**.

Step 19: Configure Smoke Tests

The Smoke Tests errand runs basic functionality tests against your Pivotal Application Service (PAS) deployment after an installation or update. In this section, choose where to run smoke tests. In the **Errands** section, you can choose whether or not to run the Smoke Tests errand.

1. Select **Smoke Tests**.

2. If you have a shared apps domain, select **Temporary space within the system organization**, which creates a temporary space within the `system` organization for running smoke tests and deletes the space afterwards. Otherwise, select **Specified org and space** and complete the fields to specify where you want to run smoke tests.



3. Click **Save**.

Step 20: (Optional) Enable Advanced Features

The **Advanced Features** section of Pivotal Application Service (PAS) includes new functionality that may have certain constraints. Although these features are fully supported, Pivotal recommends caution when using them in production environments.

Diego Cell Memory and Disk Overcommit

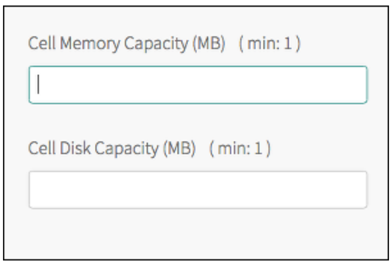
If your apps do not use the full allocation of disk space and memory set in the **Resource Config** tab, you might want use this feature. These fields control the amount to overcommit disk and memory resources to each Diego Cell VM.

For example, you might want to use the overcommit if your apps use a small amount of disk and memory capacity compared to the amounts set in the **Resource Config** settings for **Diego Cell**.

> **Note**: Due to the risk of app failure and the deployment-specific nature of disk and memory use, Pivotal has no recommendation about how much, if any, memory or disk space to overcommit.

To enable overcommit, follow these steps:

1. Select **Advanced Features**.



2. Enter the total desired amount of Diego cell memory value in the **Cell Memory Capacity (MB)** field. Refer to the **Diego Cell** row in the **Resource Config** tab for the current Cell memory capacity settings that this field overrides.

3. Enter the total desired amount of Diego cell disk capacity value in the **Cell Disk Capacity (MB)** field. Refer to the **Diego Cell** row in the **Resource Config** tab for the current Cell disk capacity settings that this field overrides.

4. Click **Save**.

> **Note**: Entries made to each of these two fields set the total amount of resources allocated, not the overage.

Whitelist for Non-RFC-1918 Private Networks

Some private networks require extra configuration so that internal file storage (WebDAV) can communicate with other PCF processes.

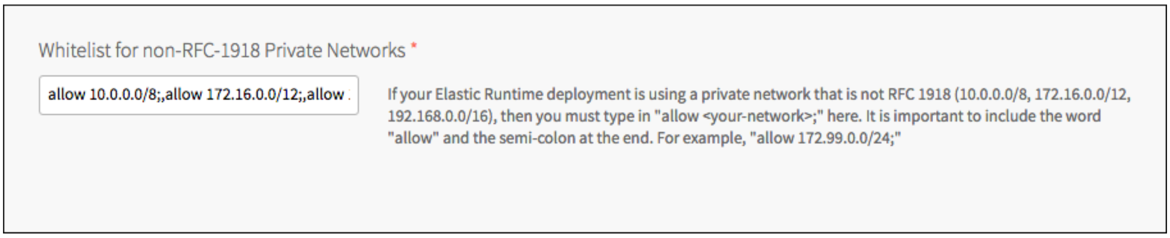
The **Whitelist for non-RFC-1918 Private Networks** field is provided for deployments that use a non-RFC 1918 private network. This is typically a private network other than `10.0.0.0/8`, `172.16.0.0/12`, or `192.168.0.0/16`.

Most PCF deployments do not require any modifications to this field.

To add your private network to the whitelist, perform the following steps:

1. Select **Advanced Features**.

2. Append a new `allow` rule to the existing contents of the **Whitelist for non-RFC-1918 Private Networks** field.



Include the word `allow`, the network CIDR range to allow, and a semi-colon (`;`) at the end. For example: `allow 172.99.0.0/24;`

3. Click **Save**.

CF CLI Connection Timeout

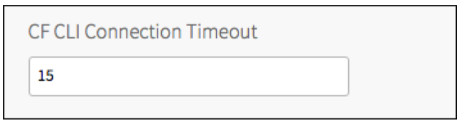
The **CF CLI Connection Timeout** field allows you to override the default five second timeout of the Cloud Foundry Command Line Interface (cf CLI) used within your PCF deployment. This timeout affects the cf CLI command used to push PAS errand apps such as Notifications, Autoscaler, and Apps Manager.

Set the value of this field to a higher value, in seconds, if you are experiencing domain name resolution timeouts when pushing errands in PAS.

To modify the value of the **CF CLI Connection Timeout**, perform the following steps:

1. Select **Advanced Features**.

2. Add a value, in seconds, to the **CF CLI Connection Timeout** field.



3. Click **Save**.

Step 21: Configure Errands

Errands are scripts that Ops Manager runs automatically when it installs or uninstalls a product, such as a new version of Pivotal Application Service (PAS). There are two types of errands: *post-deploy errands* run after the product is installed, and *pre-delete errands* run before the product in uninstalled.

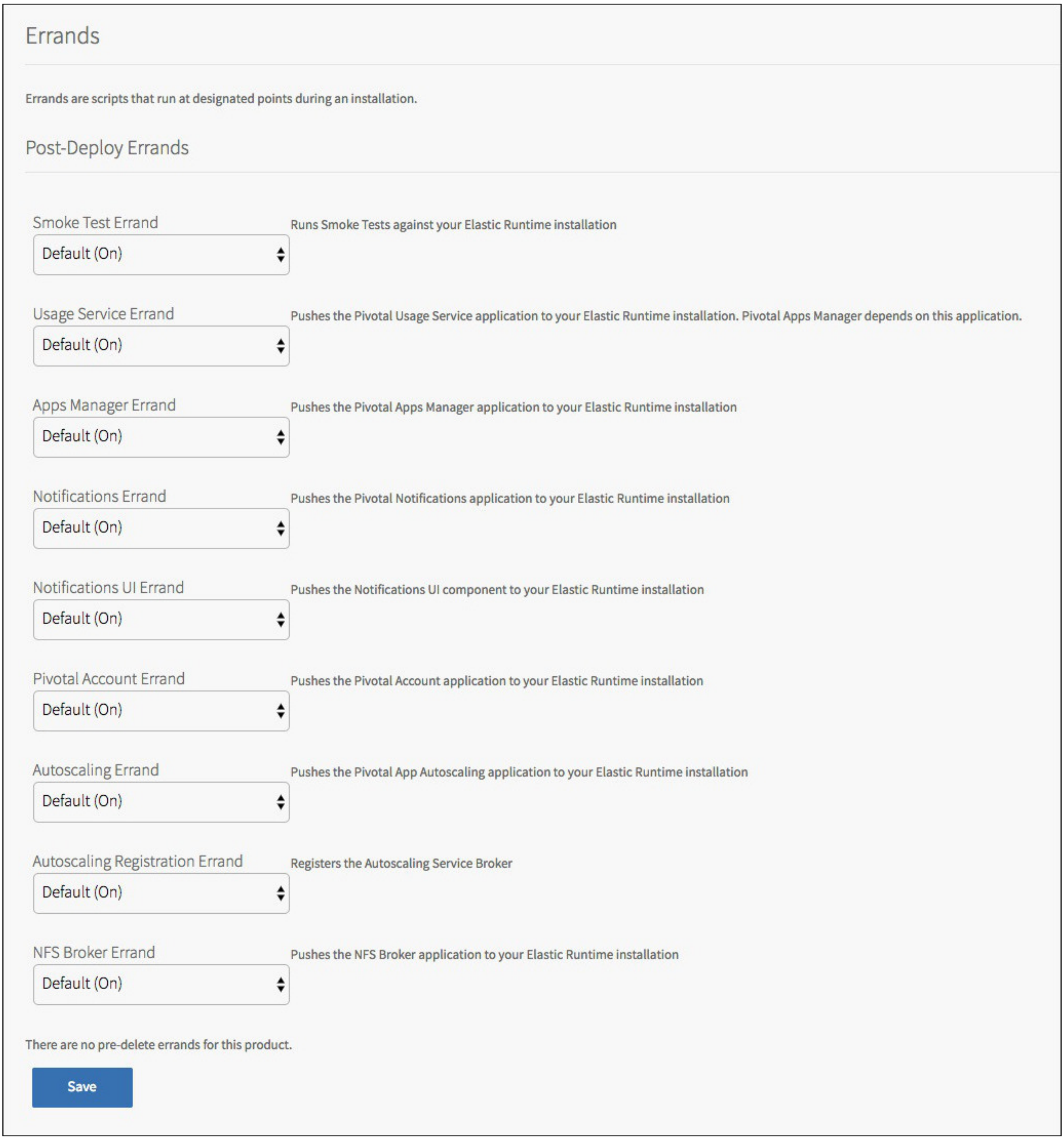
By default, Ops Manager always runs pre-delete errands, and only runs post-deploy errands when the product has changed since the last time Ops

Manager installed something. In PAS, the Smoke Test Errand defaults to always run.

The PAS tile **Errands** pane lets you change these run rules. For each errand, you can select **On** to run it always, **Off** to never run it, or **When Changed** to run it only when the product has changed since the last install.

For more information about how Ops Manager manages errands, see the [Managing Errands in Ops Manager]() topic.

> **Note:** Several errands deploy apps that provide services for your deployment, such as Autoscaling and Notifications. Once one of these apps is running, selecting **Off** for the corresponding errand on a subsequent installation does not stop the app.



- **Smoke Test Errand** verifies that your deployment can do the following:
  - Push, scale, and delete apps
  - Create and delete orgs and spaces
- **Usage Service Errand** deploys the Pivotal Usage Service application, which Apps Manager depends on.
- **Apps Manager Errand** deploys Apps Manager, a dashboard for managing apps, services, orgs, users, and spaces. Until you deploy Apps Manager, you

must perform these functions through the cf CLI. After Apps Manager has been deployed, Pivotal recommends deselecting the checkbox for this errand on subsequent PAS deployments. For more information about Apps Manager, see the Getting Started with the Apps Manager topic.

- **Notifications Errand** deploys an API for sending email notifications to your PCF platform users.

> **Note**: The Notifications app requires that you configure SMTP with a username and password, even if you set the value of **SMTP Authentication Mechanism** to `none`.

- **Notifications UI Errand** deploys a dashboard for users to manage notification subscriptions.
- **Pivotal Account Errand** deploys Pivotal Account, a dashboard that allows users to create and manage their accounts. In the Pivotal Account dashboard, users can launch applications, manage their profiles, manage account security, manage notifications, and manage approvals. See the Enabling Pivotal Account topic for more information.
- **Autoscaling Errand** enables you to configure your apps to automatically scale in response to changes in their usage load. See the Scaling an Application Using Autoscaler topic for more information.
- **Autoscaling Registration Errand** makes the Autoscaling service available to your applications. Without this errand, you cannot bind the Autoscaling app to your apps.
- **NFS Broker Errand** enables you to use NFS Volume Services by installing the NFS Broker app in PAS. See the Enabling NFS Volume Services topic for more information.

Step 22: Configure Router (or HAProxy) to Elastic Load Balancer

1. In the PAS tile, click **Resource Config**.
2. Enter the name of you SSH load balancer depending on which release you are using. You can specify multiple load balancers by entering the names separated by commas.
   - **Pivotal Application Service (PAS)**: In the **ELB Name** field of the **Diego Brain** row, enter the value of `ssh_elb_name` from the terraform output.
   - **Small Footprint Runtime**: In the **ELB Name** field of the **Control** row, enter the value of `ssh_elb_name` from the terraform output.
3. In the **ELB Name** field of the **Router** row, enter the value of `web_elb_name` from the terraform otuput.

> **Note**: If you are using HAProxy in your deployment, then put the name of the load balancers in the **ELB Name** field of the **HAProxy** row instead of the **Router** row. For a high availability configuration, scale up the HAProxy job to more than one instance.

4. In the **ELB Name** field of the **TCP Router** row, enter the value of `tcp_elb_name` from the terraform output.

5. Click **Save**.

Step 23: (Optional) Scale Down and Disable Resources

> **Note**: The **Resource Config** pane has fewer VMs if you are installing the Small Footprint Runtime.

> **Note**: The Small Footprint Runtime does not default to a highly available configuration. It defaults to the minimum configuration. If you want to make the Small Footprint Runtime highly available, scale the **Compute**, **Router**, and **Database** VMs to `3` instances and scale the **Control** VM to `2` instances.

Pivotal Application Service (PAS) defaults to a highly available resource configuration. However, you may need to perform additional procedures to make your deployment highly available. See the Zero Downtime Deployment and Scaling in CF and the Scaling Instances in PAS topics for more information.

If you do not want a highly available resource configuration, you must scale down your instances manually by navigating to the **Resource Config** section and using the drop-down menus under **Instances** for each job.

By default, PAS also uses an internal filestore and internal databases. If you configure PAS to use external resources, you can disable the corresponding system-provided resources in Ops Manager to reduce costs and administrative overhead.

Complete the following procedures to disable specific VMs in Ops Manager:

1. Click **Resource Config**.

2. If you configure PAS to use an external S3-compatible filestore, edit the following fields:
   - **File Storage**: Enter `0` in **Instances**.

3. If you selected **External** when configuring the UAA or System databases, edit the following fields:
   - **MySQL Proxy**: Enter `0` in **Instances**.
   - **MySQL Server**: Enter `0` in **Instances**.
   - **MySQL Monitor**: Enter `0` in **Instances**.
   - **Cloud Controller Database**: Enter `0` in **Instances**.
   - **UAA Database**: Enter `0` in **Instances**.

4. If you are not using HAProxy, enter `0` in the **Instances** field for **HAProxy**.

5. Click **Save**.

Step 24: Download Stemcell

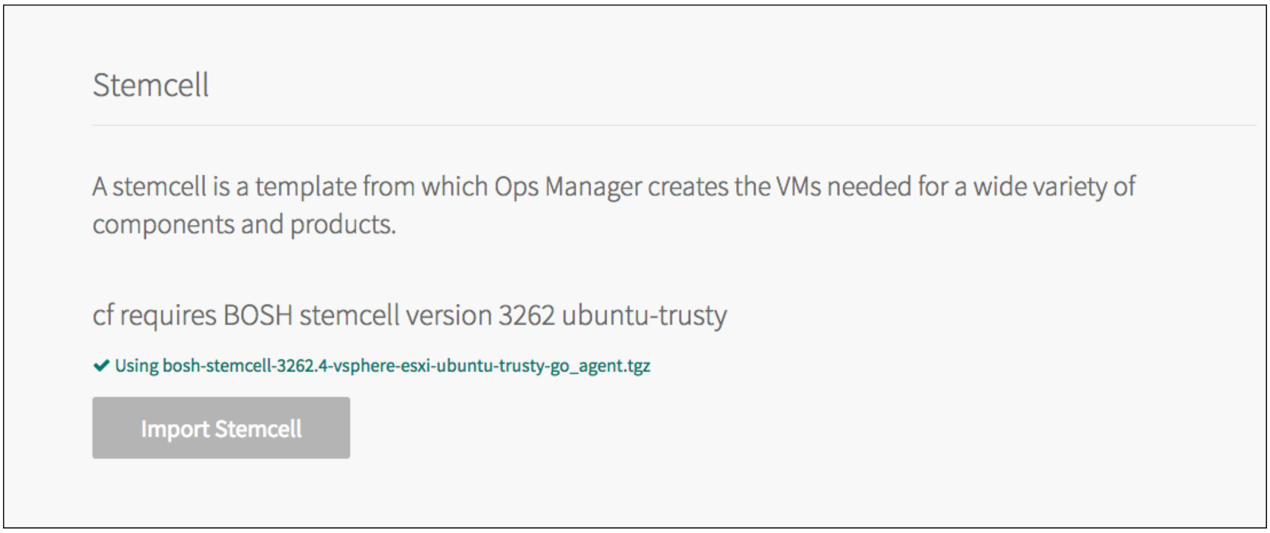
This step is only required if your Ops Manager does not already have the Stemcell version required by PAS.

1. Select **Stemcell**.

2. Log into the Pivotal Network and click on **Stemcells**.

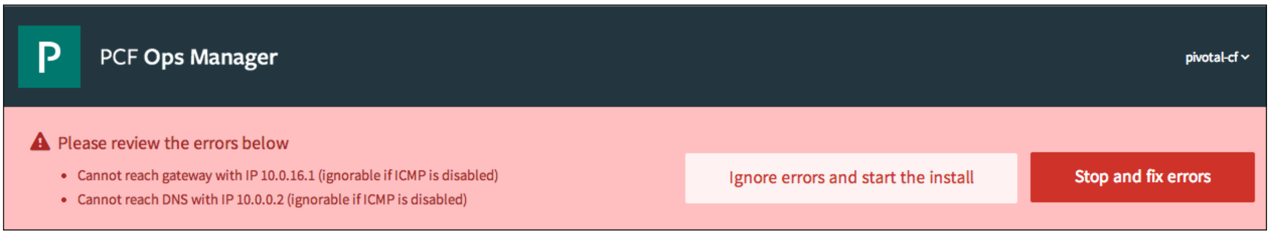
3. Download the appropriate stemcell version targeted for your IaaS.

4. In Ops Manager, import the downloaded stemcell `.tgz` file.

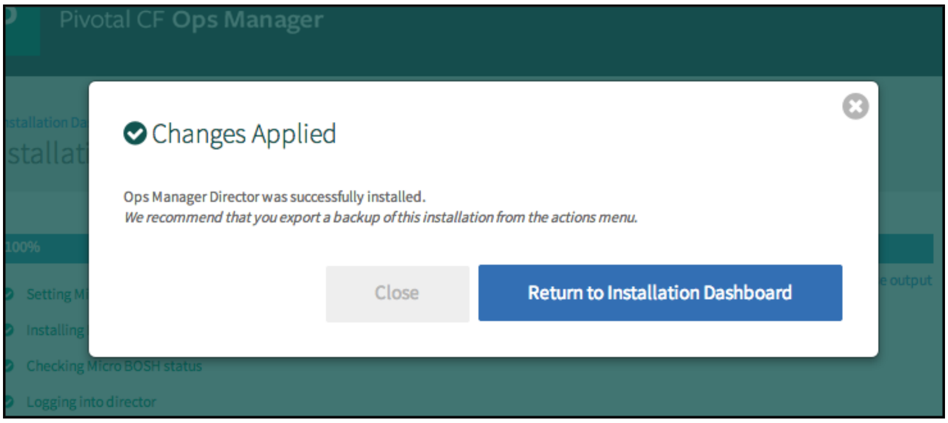


Step 25: Complete the PAS Installation

1. Click the **Installation Dashboard** link to return to the Installation Dashboard.
2. Click **Apply Changes**. If the following ICMP error message appears, click **Ignore errors and start the install**.



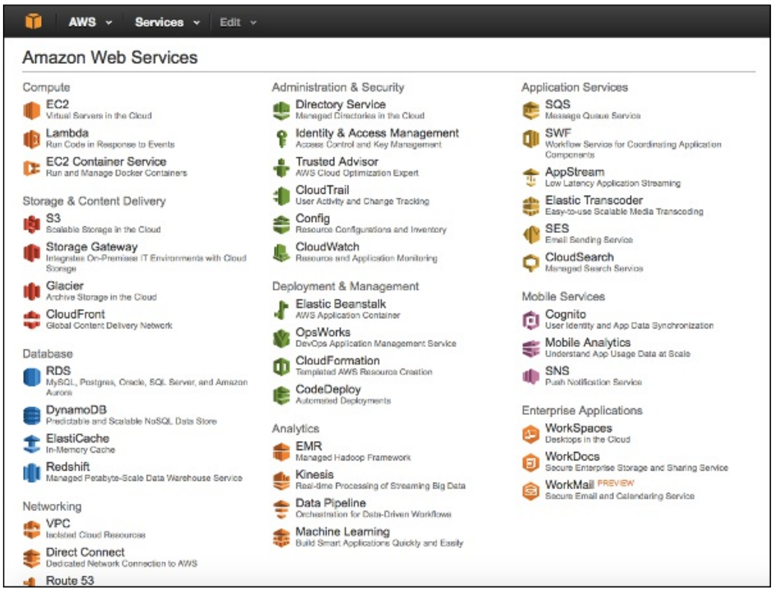
The install process generally requires a minimum of 90 minutes to complete. The image shows the Changes Applied window that displays when the installation process successfully completes.



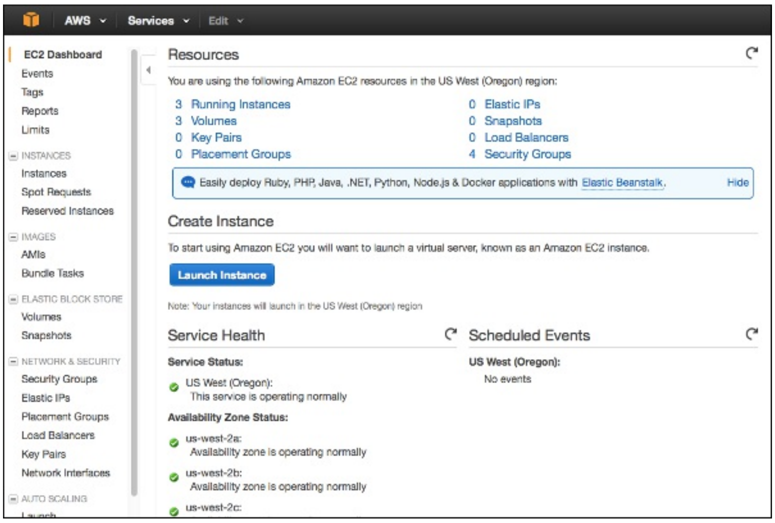
Deleting PCF from AWS

Page last updated:

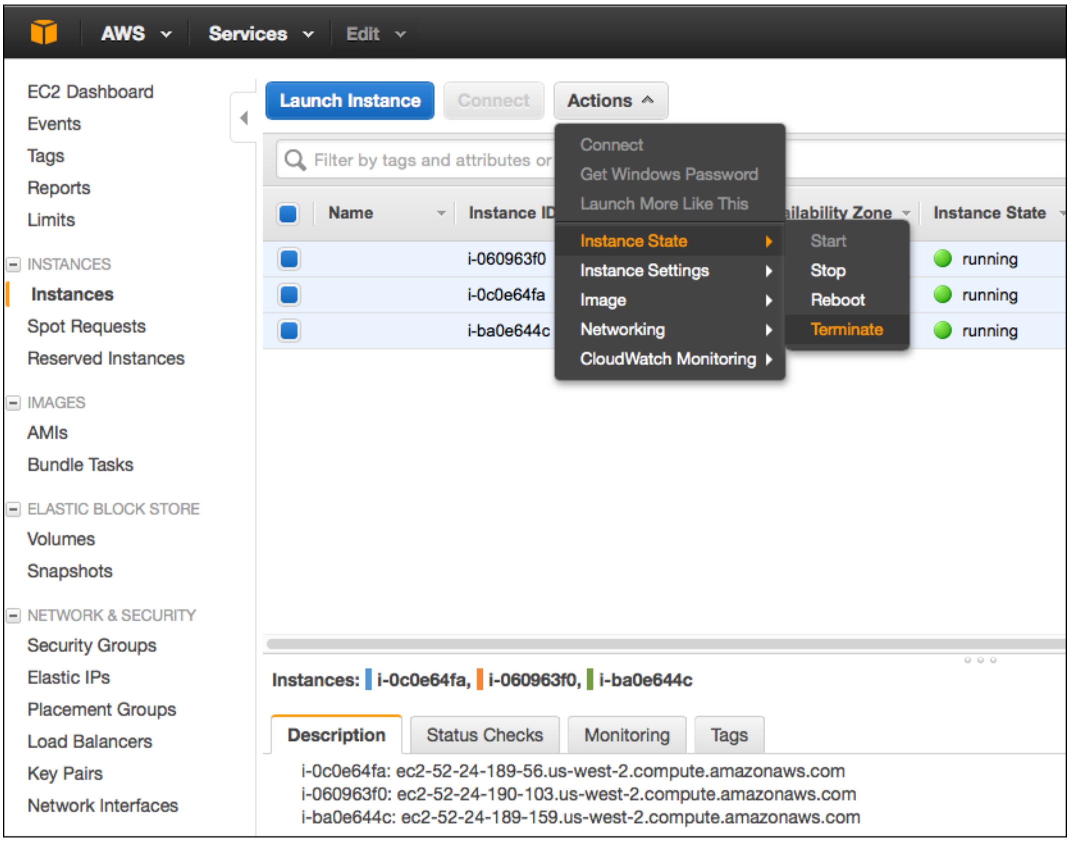
When you deploy [Pivotal Cloud Foundry](#) (PCF) to Amazon Web Services (AWS), you provision a set of resources. This topic describes how to delete the AWS resources associated with a PCF deployment. You can use the AWS console to remove an installation of all components, but retain the objects in your bucket for a future deployment.

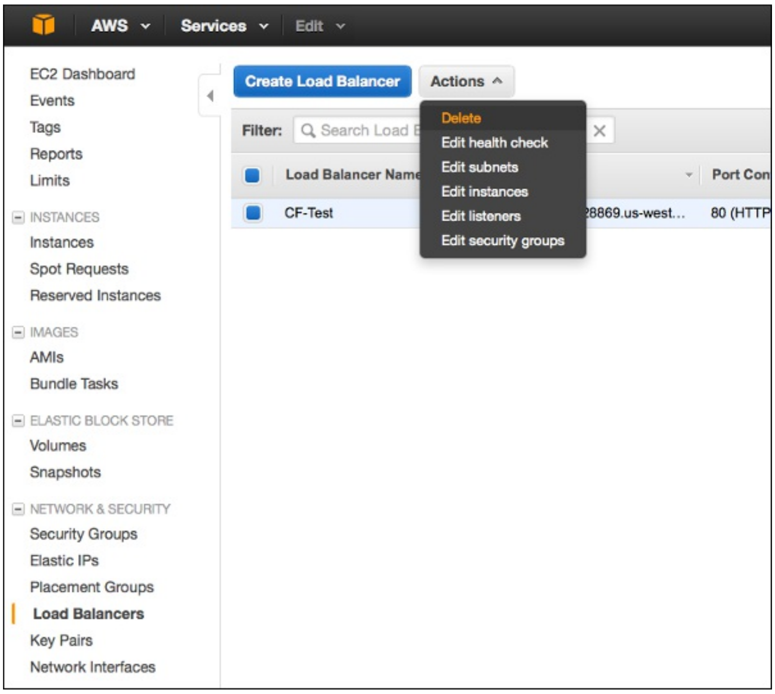
1. Log into your AWS Console. 

2. Navigate to your EC2 dashboard. Select **Instances** from the menu on the left side.

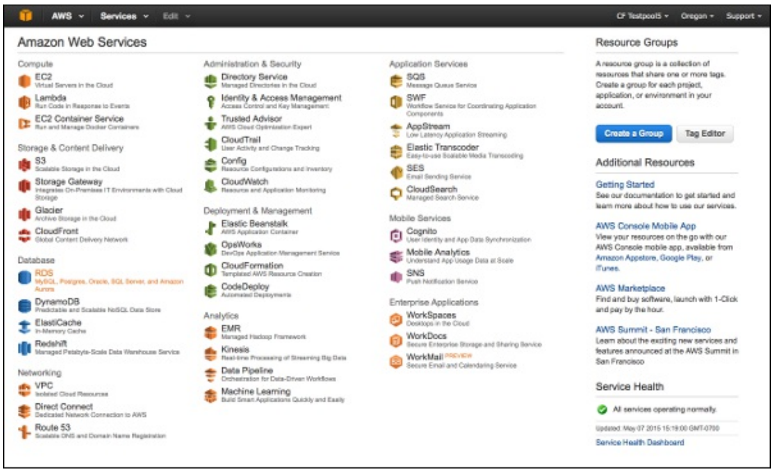


3. Terminate all your instances.



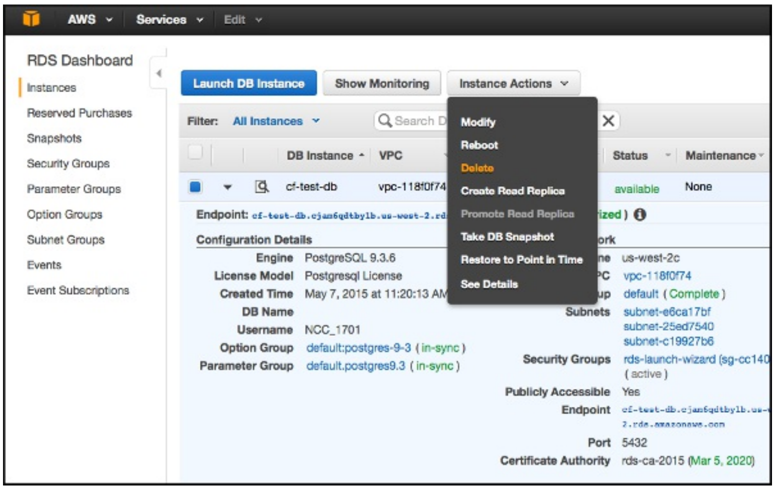


4. Select **Load Balancers**. Delete all load balancers.

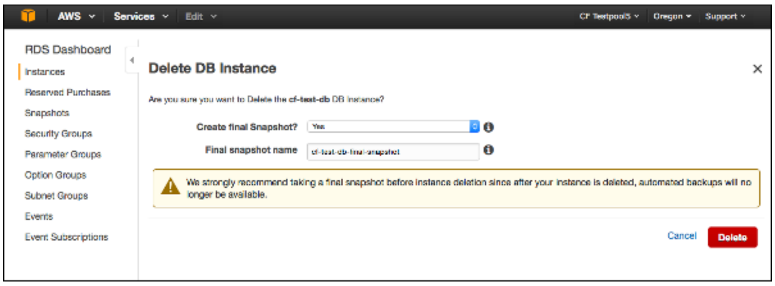


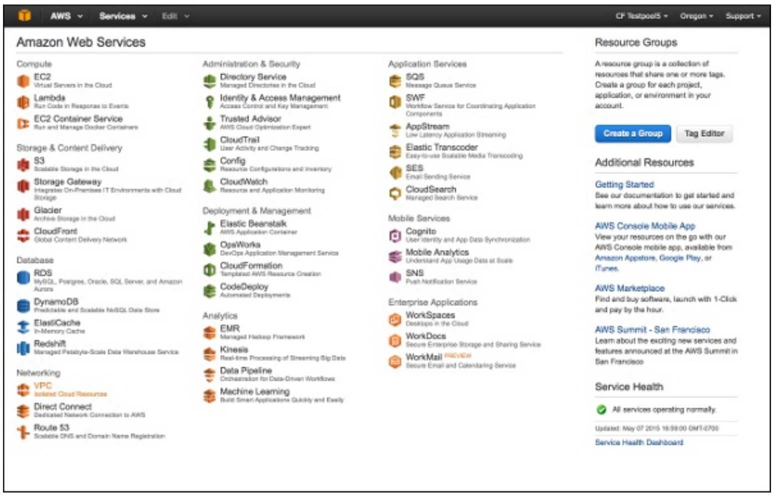
5. From the AWS Console, select **RDS**.

6. Select **Instances** from the menu on the left side. Delete the RDS instances.



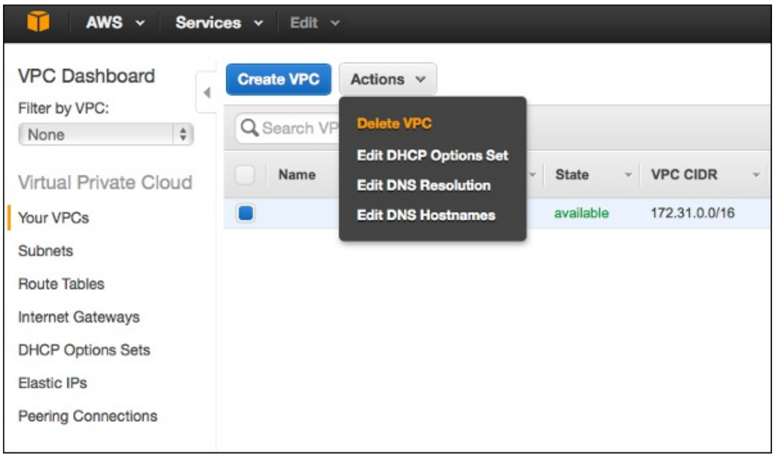
7. Select **Create final Snapshot** from the drop-down menu. Click **Delete**.



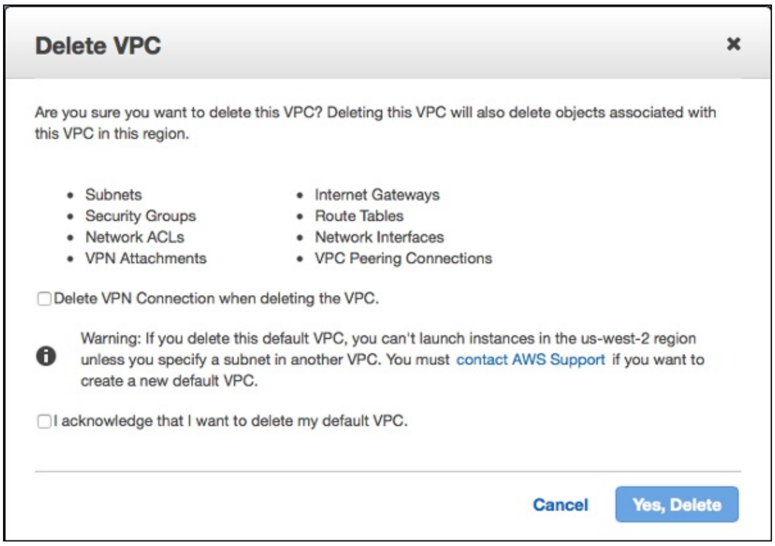


8. From the AWS Console, select **VPC**.

9. Select **Your VPCs** from the menu on the left. Delete the VPCs.



10. Check the box to acknowledge that you want to delete your default VPC. Click **Yes, Delete**.



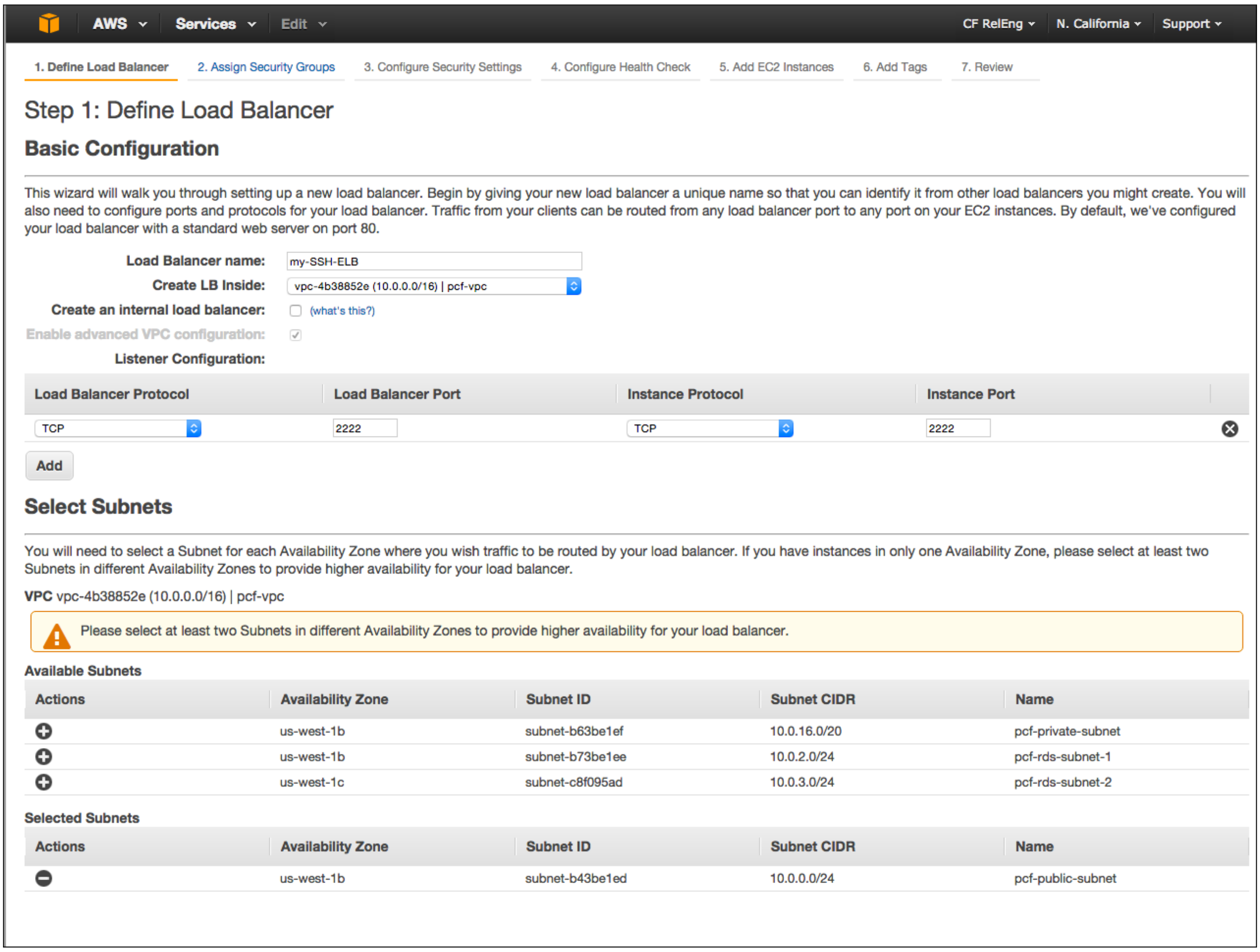
Creating a Proxy ELB for Diego SSH

If you want to allow SSH connections to application containers, you may want to use an Elastic Load Balancer (ELB) as the SSH proxy.

Perform the steps below to create this ELB:

1. On the EC2 Dashboard, click **Load Balancers**.

2. Click **Create Load Balancer**, and configure a load balancer with the following information:

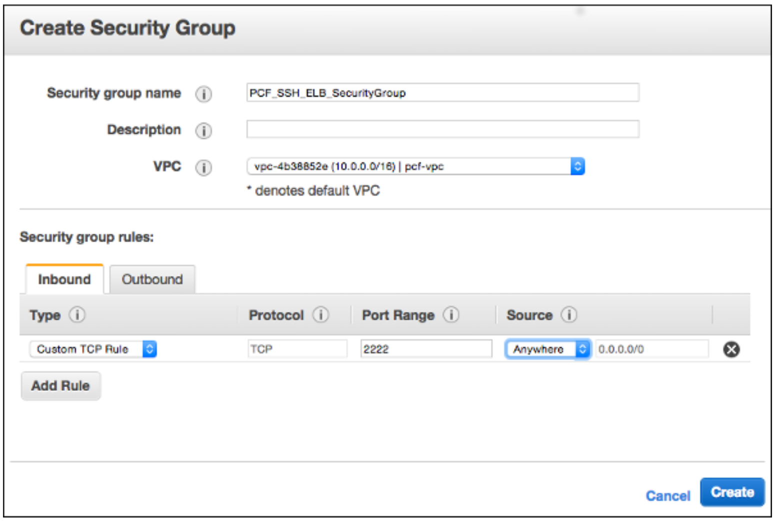


   - Enter a load balancer name.
   - **Create LB Inside**: Select the **pcf-vpc** VPC where your PCF installation lives.
   - Ensure that the **Create an internal load balancer** checkbox is not selected.

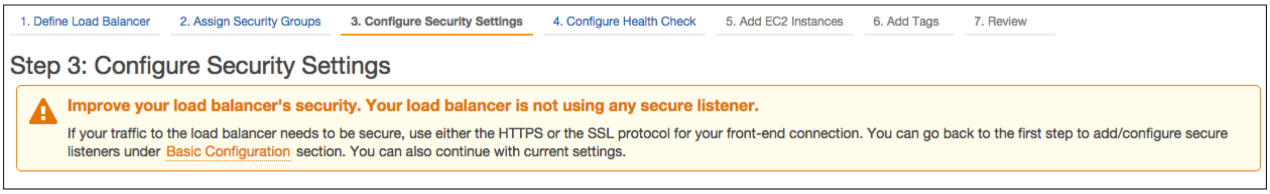
3. Under **Load Balancer Protocol**, ensure that this ELB is listening on TCP port `2222` and forwarding to TCP port `2222`.

4. Under **Select Subnets**, select the public subnet.

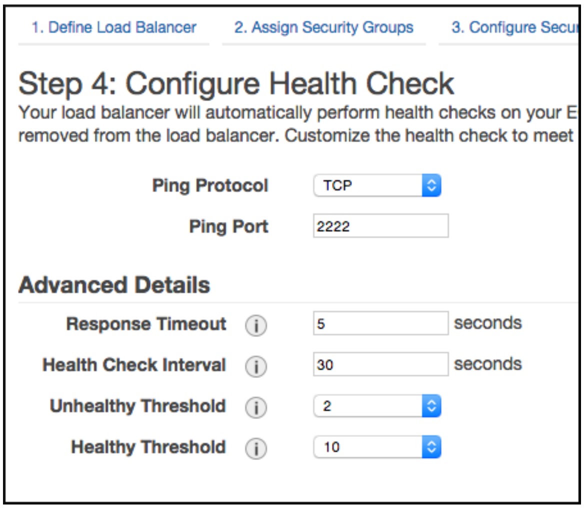
5. On the **Assign Security Groups** page, create a new Security Group. This Security Group should allow inbound traffic on TCP port `2222`.



6. The **Configure Security Settings** page displays a security warning because your load balancer is not using a secure listener. You can ignore this warning.



7. Click **Next: Configure Health Check**.



8. Select **TCP in Ping Protocol** on the **Configure Health Check** page. Ensure that the **Ping Port** value is `2222` and set the **Health Check Interval** to `30` seconds.

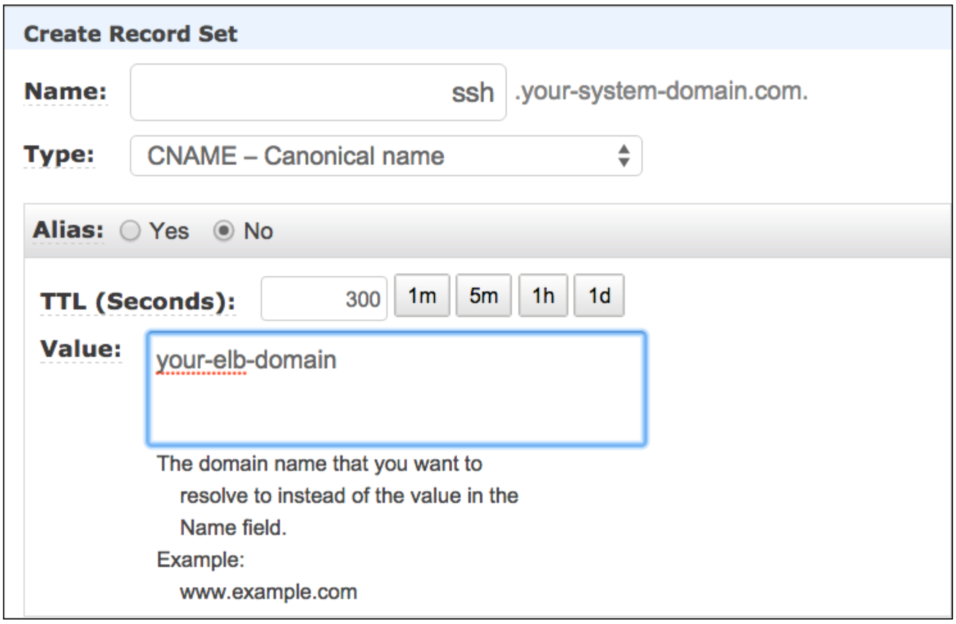
9. Click **Next: Add EC2 Instances**.

10. Accept the defaults on the **Add EC2 Instances** page and click **Next: Add Tags**.

11. Accept the defaults on the **Add Tags** page and click **Review and Create**.

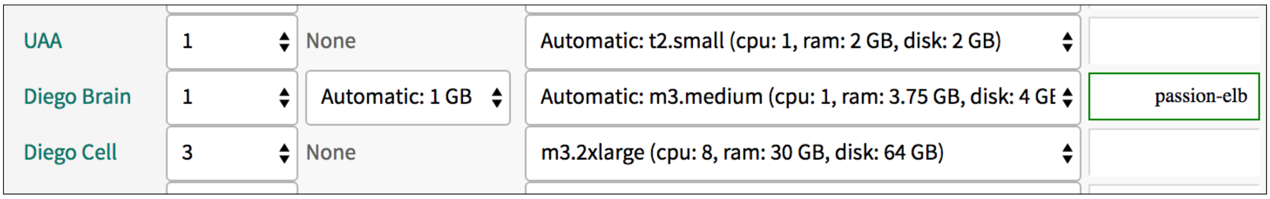
12. Review and confirm the load balancer details, and click **Create**.

13. With your DNS service (for example, Amazon Route 53), create an `ssh.system.YOUR-SYSTEM-DOMAIN` DNS record that points to this ELB that you just created.



14. You can now use this ELB to the SSH Proxy of your Pivotal Application Service (PAS) installation.

15. In PAS, select **Resource Config**, and enter the ELB that you just created in the **Diego Brain** row, under the ELB Names column.



PCF on Azure Requirements

Page last updated:

This guide describes how to install Pivotal Cloud Foundry (PCF) on Azure.

To view production-level deployment options for PCF on Azure, see the Reference Architecture for Pivotal Cloud Foundry on Azure.

General Requirements

The following are general requirements for deploying and managing a PCF deployment with Ops Manager and Pivotal Application Service (PAS):

- A wildcard DNS record that points to your router or load balancer. Alternatively, you can use a service such as xip.io. For example, `203.0.113.0.xip.io`.

  PAS gives each application its own hostname in your app domain. With a wildcard DNS record, every hostname in your domain resolves to the IP address of your router or load balancer, and you do not need to configure an A record for each app hostname. For example, if you create a DNS record `*.example.com` pointing to your load balancer or router, every application deployed to the `example.com` domain resolves to the IP address of your router.
- At least one wildcard TLS certificate that matches the DNS record you set up above, `*.example.com`.
- Sufficient IP allocation:
  - One static IP address for either HAProxy or one of your gorouters
  - One IP address for each VM instance
  - An additional IP address for each compilation worker
    So the formula for total IPs needed is `IPs needed = static IPs + VM instances + compilation workers`

  > **Note**: Pivotal recommends that you allocate at least 36 dynamic IP addresses when deploying Ops Manager and PAS. BOSH requires additional dynamic IP addresses during installation to compile and deploy VMs, install PAS, and connect to services.

- One or more NTP servers if not already provided by your IaaS.
- (**Recommended**) A network without DHCP available for deploying the PAS VMs.

  > **Note**: If you have DHCP, refer to the Troubleshooting Guide to avoid issues with your installation.

- (**Optional**) External storage. When you deploy PCF, you can select internal file storage or external file storage, either network-accessible or IaaS-provided, as an option in the PAS tile. Pivotal recommends using external storage whenever possible. See Upgrade Considerations for Selecting File Storage in Pivotal Cloud Foundry for a discussion of how file storage location affects platform performance and stability during upgrades.
- (**Optional**) External databases. When you deploy PCF, you can select internal or external databases for the BOSH Director and for PAS. Pivotal recommends using external databases in production deployments.
- (**Optional**) External user stores. When you deploy PCF, you can select a SAML user store for Ops Manager or a SAML or LDAP user store for PAS, to integrate existing user accounts.
- The most recent version of the Cloud Foundry Command Line Interface (cf CLI).

Azure Requirements

The following are the minimum resource requirements for deploying a PCF deployment with Ops Manager and Pivotal Application Service (PAS) on Azure:

- VMs:
  - 27 F1s
  - 4 F2s
  - 4 F4s
  - 1 DS11v2
  - 1 DS12v2

  > **Note**: Specific instance types are only supported in certain regions. See the Azure documentation for a complete list. If you are deploying PCF in a region that does not support the above instance types, see the Ops Manager API documentation at `https://YOUR-OPS-MANAGER-FQDN/docs#overriding-defaults-with-custom-disk-types` for instructions on how to override the default VM sizes. Changing the default VM sizes may increase the cost of your deployment.

By default, PAS deploys the number of VM instances required to run a highly available configuration of PCF. If you are deploying a test or sandbox PCF that does not require HA, then you can scale down the number of instances in your deployment. For information about the number of instances required to run a minimal, non-HA PCF deployment, see Scaling PAS.

- An OS disk of 120 GB for the Ops Manager VM

> **Note:** Ops Manager v1.11 requires a Director VM with at least 8 GB memory.

- To deploy PCF on Azure, you must have the Azure CLI v2.0. For instructions on how to install the Azure CLI for your operating system, see Preparing to Deploy PCF on Azure.

Azure Permissions Guidelines

Pivotal recommends following the principle of least privilege by scoping privileges to the most restrictive permissions possible for a given role. See IaaS Permissions Guidelines for recommendations on how to create and scope Azure accounts for PCF.

Azure Security Documents

- Azure security documentation
  This site has documentation on Azure security tools. It provides a general guide to how to manage IaaS users and credentials.

Install PCF on Azure

Complete the following procedures to install PCF on Azure:

1. Preparing to Deploy PCF on Azure
2. You can choose to deploy Ops Manager Director with an Azure Resource Manager (ARM) template, or manually:
   - Launching an Ops Manager Director Instance with an ARM Template (Recommended)
   - Launching an Ops Manager Director Instance on Azure without an ARM Template
3. Configuring Ops Manager Director on Azure
4. Deploying PAS on Azure

Install PCF on Azure Government Cloud

> **Note:** Azure Government Cloud is only supported in PCF 1.10 and later.

To deploy PCF on Azure Government Cloud, see the Deploying PCF on Azure Government Cloud topic.

Install PCF in Azure Germany

> **Note:** Azure Germany is only supported in PCF 1.10.1 and later.

To deploy PCF in Azure Germany, see the Deploying PCF in Azure Germany topic.

Troubleshoot PCF on Azure

To troubleshoot known issues when deploying PCF on Azure, see the Troubleshooting PCF on Azure topic.

Delete PCF on Azure

You can use the Azure Portal console to remove all the components of a PCF on Azure installation.

- Deleting a PCF on Azure Installation

Upgrade PCF on Azure

Upgrade your Ops Manager Director as part of the upgrade process documented in the Upgrading Pivotal Cloud Foundry topic.

- Upgrading Ops Manager Director on Azure

Preparing Azure

Page last updated:

This topic describes how to prepare to deploy Pivotal Cloud Foundry (PCF) on Azure by creating a service principal to access resources in your Azure subscription.

After you complete this procedure, follow the instructions in either the Launching an Ops Manager Director Instance with an ARM Template topic or the Launching an Ops Manager Director Instance on Azure without an ARM Template topic.

Step 1: Install and Configure the Azure CLI

1. Install the Azure CLI 2.0 by following the instructions for your operating system in the Azure documentation.

2. Set your cloud with the `--name` value corresponding to the Azure environment you are installing PCF on:
   - **Azure:** `AzureCloud`
   - **Azure China:** `AzureChinaCloud`
   - **Azure Government Cloud:** `AzureUSGovernment`
   - **Azure Germany:** `AzureGermanCloud`

   ```
   $ az cloud set --name AzureCloud
   ```

   Note: If logging in to `AzureChinaCloud` fails with a `CERT_UNTRUSTED` error, use the latest version of node, 4.x or later.

   Note: Azure Government Cloud is only supported in PCF 1.10 and later.

3. Log in:

   ```
   $ az login
   ```

   Authenticate by navigating to the URL in the output, entering the provided code, and clicking your account.

Step 2: Set Your Default Subscription

1. Run `az account list` to list your Azure subscriptions:

   ```
   $ az account list
   [
    {
     "id": "12345678-1234-5678-1234-567891234567",
     "name": "Sample Subscription",
     "user": {
       "name": "Sample Account",
       "type": "user"
     },
     "tenantId": "11111111-1234-5678-1234-678912345678",
     "state": "Enabled",
     "isDefault": true,
     "registeredProviders": [],
     "environmentName": "AzureCloud"
   },
   {
     "id": "87654321-1234-5678-1234-678912345678",
     "name": "Sample Subscription1",
     "user": {
       "name": "Sample Account1",
       "type": "user"
     },
     "tenantId": "22222222-1234-5678-1234-678912345678",
     "state": "Enabled",
     "isDefault": false,
     "registeredProviders": [],
     "environmentName": "AzureCloud"
    }
   ]
   ```

2. Locate your default subscription by finding the subscription with `isDefault` set to `true`. If your default subscription is not where you want to deploy PCF, run `az account set --subscription SUBSCRIPTION_ID` to set a new default, where `SUBSCRIPTION_ID` is the value of the `id` field. For example: `"87654321-1234-5678-1234-567891234567"`.

```
$ az account set --subscription SUBSCRIPTION_ID
```

3. Record the value of the `id` set as the default. You use this value in future configuration steps.

4. Record the value of `tenantID` for your default subscription. This is your `TENANT_ID` for creating a service principal. If your `tenantID` value is not defined, you may be using a personal account to log in to your Azure subscription.

Step 3: Create an Azure Active Directory (AAD) Application

1. Run the following command to create an AAD application, replacing `PASSWORD` with a password of your choice. This is your `CLIENT_SECRET` for creating a service principal.

```
$ az ad app create --display-name "Service Principal for BOSH" \
--password "PASSWORD" --homepage "http://BOSHAzureCPI" \
--identifier-uris "http://BOSHAzureCPI"
```

> **Note**: You can provide any string for the `home-page` and `identifier-uris` flags, but the value of `identifer-uris` must be unique within the organization associated with your Azure subscription. For the `home-page`, Pivotal recommends using `http://BOSHAzureCPI`, as shown in the example above.

2. Record the value of `appId` from the output. This is your `APPLICATION_ID` for creating a service principal.

```
{
 "appId": "5c552e8f-b977-45f5-a50b-981cfe17cb9d",
 "appPermissions": null,
 "availableToOtherTenants": false,
 "displayName": "Service Principal for BOSH",
 "homepage": "http://BOSHAzureCIP",
 "identifierUris": [
  "http://BOSHAzureCPI"
 ],
 "objectId": "f3884df4-7d1d-4894-a78c-c1fe75750436",
 "objectType": "Application",
 "replyUrls": []
}
```

Step 4: Create and Configure a Service Principal

1. To create a service principal, run `az ad sp create --id YOUR-APPLICATION-ID`, replacing `YOUR-APPLICATION-ID` with the `APPLICATION_ID` you recorded in the previous step:

```
$ az ad sp create --id YOUR-APPLICATION-ID
{
 "appId": "5c552e8f-b977-45f5-a50b-981cfe17cb9d",
 "displayName": "Service Principal for BOSH",
 "objectId": "cc13c685-4c3b-461e-ae96-7a0563960b83",
 "objectType": "ServicePrincipal",
 "servicePrincipalNames": [
  "5c552e8f-b977-45f5-a50b-981cfe17cb9d",
  "http://BOSHAzureCPI"
 ]
}
```

2. You must have the Contributor role on your service principal to deploy PCF. Run the following command to assign this role:

```
$ az role assignment create --assignee "SERVICE-PRINCIPAL-NAME" \
--role "Contributor" --scope /subscriptions/SUBSCRIPTION-ID
```

   - For `SERVICE-PRINCIPAL-NAME`, use any value of `Service Principal Names` from the output above, such as `YOUR-APPLICATION-ID`.
   - For `SUBSCRIPTION-ID`, use the ID of the default subscription that you recorded in Step 2.

> **Note:** If you need to use multiple resource groups for your PCF deployment on Azure, you can define custom roles for your Service Principal. These roles allow BOSH to deploy PCF to pre-existing network resources outside the PCF resource group. For more information, see Reference Architecture for Pivotal Cloud Foundry on Azure.

For more information about Azure Role-Based Access Control, refer to the RBAC: Built-in roles topic in the Azure documentation.

3. Verify the assignment by running the following command:

```
$ az role assignment list --assignee "SERVICE-PRINCIPAL-NAME"
[
  {
   "id": "/subscriptions/995b7eed-77ef-45ff-a5c9-1a405ffb8243/providers/Microsoft.Authorization/roleAssignments/32e644cf-ba1a-4f43-bf7c-68bf4583e463",
   "name": "32e644cf-ba1a-4f43-bf7c-68bf4583e463",
   "properties": {
    "principalId": "cc13c685-4c3b-461e-ae96-7a0563960b83",
    "principalName": "http://BOSHAzureCPI",
    "roleDefinitionId": "/subscriptions/995b7eed-77ef-45ff-a5c9-1a405ffb8243/providers/Microsoft.Authorization/roleDefinitions/b24988ac-6180-42a0-ab88-20f7382dd24c",
    "roleDefinitionName": "Contributor",
    "scope": "/subscriptions/995b7eed-77ef-45ff-a5c9-1a405ffb8243"
   },
   "type": "Microsoft.Authorization/roleAssignments"
  }
]
```

Step 5: Verify Your Service Principal

To verify your service principal, log in to your service principal with your `APPLICATION_ID`, `CLIENT_SECRET`, and `TENANT_ID`.

```
$ az login --username APPLICATION_ID --password CLIENT_SECRET \
--service-principal --tenant TENANT_ID
[
  {
   "cloudName": "AzureCloud",
   "id": "995b7eed-77ef-45ff-a5c9-1a405ffb8243",
   "isDefault": true,
   "name": "CF-Docs",
   "state": "Enabled",
   "tenantId": "29248f74-371f-4db2-9a50-c62a6877a0c1",
   "user": {
    "name": "5c552e8f-b977-45f5-a50b-981cfe17cb9d",
    "type": "servicePrincipal"
   }
  }
]
```

If you cannot log in, the service principal is invalid. Create a new service principal and try again.

Step 6: Perform Registrations

1. Register your subscription with Microsoft.Storage:

```
$ az provider register --namespace Microsoft.Storage
```

2. Register your subscription with Microsoft.Network:

```
$ az provider register --namespace Microsoft.Network
```

3. Register your subscription with Microsoft.Compute:

```
$ az provider register --namespace Microsoft.Compute
```

After you complete this topic, continue to one of the following topics:

- Launching an Ops Manager Director Instance with an ARM Template: Perform the procedures in this topic to deploy Ops Manager Director with an Azure Resource Manager (ARM) template. Pivotal recommends using an ARM template.

- [Launching an Ops Manager Director Instance on Azure without an ARM Template](): Perform the procedures in this topic to deploy Ops Manager Director manually.

Deploying BOSH and Ops Manager to Azure with ARM

Page last updated:

This topic describes how to deploy Ops Manager Director for Pivotal Cloud Foundry (PCF) on Azure using an Azure Resource Manager (ARM) template. An ARM template is a JSON file that describes one or more resources to deploy to a resource group.

You can also deploy Ops Manager Director manually, by following the procedure in the Deploying BOSH and Ops Manager to Azure Manually. Manual deployment is required if you are deploying to:

- Azure China
- Azure Germany
- Azure Government Cloud
- Azure Stack

Before you perform the procedures in this topic, you must complete the procedures in the Preparing Azure topic. After you complete the procedures in this topic, follow the instructions in Configuring Ops Manager Director on Azure.

Where to Find the PCF Azure ARM Templates

The PCF Azure ARM Templates are available for download from the following GitHub repository:

https://github.com/pivotal-cf/pcf-azure-arm-templates

For PCF v1.11 and later, use the templates tagged with the 1.11+ release.

For PCF v1.11 and earlier, use the templates tagged with the 1.10- release.

Step 1: Create BOSH Storage Account

Azure for PCF uses multiple general-purpose Azure storage accounts. The BOSH and Ops Manager VMs use one main BOSH storage account, and the other components share five or more deployment storage accounts.

1. Choose a name for your resource group and export it as an environment variable `$RESOURCE_GROUP`.

   ```
   $ export RESOURCE_GROUP="YOUR-RESOURCE-GROUP-NAME"
   ```

   Note: If you are on a Windows machine, you can use `set` instead of `export`.

2. Export your location. For example, `westus`.

   ```
   $ export LOCATION="YOUR-LOCATION"
   ```

   Note: For a list of available locations, run `az account list-locations`.

3. Create your resource group:

   ```
   $ az group create --name $RESOURCE_GROUP --location $LOCATION
   ```

4. Choose a name for your BOSH storage account, and export it as the environment variable `$STORAGE_NAME`. Storage account names must be globally unique across Azure, between 3 and 24 characters in length, and contain only lowercase letters and numbers.

   ```
   $ export STORAGE_NAME="YOUR-BOSH-STORAGE-ACCOUNT-NAME"
   ```

5. Create the storage account.

```
$ az storage account create --name $STORAGE_NAME \
--resource-group $RESOURCE_GROUP --sku Standard_LRS \
--kind Storage --location $LOCATION
```

Note: `Standard_LRS` refers to a Standard Azure storage account. The BOSH Director requires table storage to store stemcell information. Azure Premium storage does not support table storage and cannot be used for the BOSH storage account.

6. Retrieve the connection string for your BOSH storage account:

```
$ az storage account show-connection-string \
--name $STORAGE_NAME --resource-group $RESOURCE_GROUP
```

The command returns output similar to the following:

```
{
 "connectionString": "DefaultEndpointsProtocol=https;EndpointSuffix=core.windows.net;AccountName=cfdocsdeploystorage1;AccountKey=EXAMPLEaaabbbcccMf8wEwdeJMvvonrb
}
```

7. Record the full value of `connectionString` from the output above, starting with and including `DefaultEndpointsProtocol=`.

8. Export the connection string:

```
$ export CONNECTION_STRING="YOUR-CONNECTION-STRING"
```

9. Create a container for the Ops Manager image:

```
$ az storage container create --name opsman-image \
--connection-string $CONNECTION_STRING
```

10. Create a container for the Ops Manager VM:

```
$ az storage container create --name vhds \
--connection-string $CONNECTION_STRING
```

11. Create a container for Ops Manager:

```
$ az storage container create --name opsmanager \
--connection-string $CONNECTION_STRING
```

12. Create a container for BOSH:

```
$ az storage container create --name bosh \
--connection-string $CONNECTION_STRING
```

13. Create a container for the stemcell:

```
$ az storage container create --name stemcell \
--public-access blob \
--connection-string $CONNECTION_STRING
```

14. Create a table for stemcell data:

```
$ az storage table create --name stemcells \
--connection-string $CONNECTION_STRING
```

Step 2: Copy Ops Manager Image

1. Navigate to Pivotal Network and download the latest release of **Pivotal Cloud Foundry Ops Manager for Azure**. You can download either a PDF or a YAML file.

2. View the downloaded file and locate the Ops Manager image URL appropriate for your region.

3. Export the Ops Manager image URL as an environment variable.

```
$ export OPS_MAN_IMAGE_URL="YOUR-OPS-MAN-IMAGE-URL"
```

4. Copy the Ops Manager image into your storage account:

```
$ az storage blob copy start --source-uri $OPS_MAN_IMAGE_URL \
--connection-string $CONNECTION_STRING \
--destination-container opsman-image \
--destination-blob image.vhd
```

5. Copying the image may take several minutes. Run the following command and examine the output under `"copy"`:

```
$ az storage blob show --name image.vhd \
--container-name opsman-image \
--account-name $STORAGE_NAME
...
"copy": {
  "completionTime": "2017-06-26T22:24:11+00:00",
  "id": "b9c8b272-a562-4574-baa6-f1a04afcefdf",
  "progress": "53687091712/53687091712",
  "source": "https://opsmanagerwestus.blob.core.windows.net/images/ops-manager-1.11.3.vhd",
  "status": "success",
  "statusDescription": null
},
```

When `status` reads `success`, continue to the next step.

Step 3: Configure the ARM Template

1. Create a keypair on your local machine. For example, enter the following command:

```
$ ssh-keygen -t rsa -f opsman -C ubuntu
```

When prompted for a passphrase, follow the prompts to provide an empty passphrase.

2. Download the latest release of the PCF Azure ARM Templates. For PCF v1.11 and later, download this release.

3. Open the parameters file and enter values for the following parameters:
   - **OpsManVHDStorageAccount**: The name of the storage account you created in Step 1: Create Storage Account
   - **BlobStorageContainer**: The name of the container to which you copied the Ops Manager VHD
   - **BlobStorageEndpoint**: The base URL of the storage endpoint. Leave the default endpoint unless you are using Azure China, Azure Government Cloud, or Azure Germany:
     - For Azure China, use `blob.core.chinacloudapi.cn`. See the Azure documentation for more information.
     - For Azure Government Cloud, use `blob.core.usgovcloudapi.net`. See the Azure documentation for more information.
     - For Azure Germany, use `blob.core.cloudapi.de`. See the Azure documentation for more information.
   - **AdminSSHKey**: The contents of the `opsman.pub` public key file that you created above
   - **Location**: The location to install the Ops Manager VM. For example, `westus`.
   - **Environment**: Tag template-created resources for assisting with resource management

Step 4: Deploy the ARM Template

1. Deploy the template:

```
$ az group deployment create --template-file azure-deploy.json \
--parameters azure-deploy-parameters.json \
--resource-group $RESOURCE_GROUP --name cfdeploy
```

2. Create a network security group named `pcf-nsg`.

```
$ az network nsg create --name pcf-nsg \
--resource-group $RESOURCE_GROUP \
--location $LOCATION
```

3. Add a network security group rule to the `pcf-nsg` group to allow traffic from the public Internet.

```
$ az network nsg rule create --name internet-to-lb \
--nsg-name pcf-nsg --resource-group $RESOURCE_GROUP \
--protocol Tcp --priority 100 \
--destination-port-range '*'
```

Step 5: Complete Ops Manager Director Configuration

1. Navigate to your DNS provider, and create an entry that points a fully qualified domain name (FQDN) in your domain to the IP address of the Ops Manager VM.

   > **Note**: If the ARM template deployment does not return an IP address in the output of `opsMan-FQDN`, then visit the Azure Portal to retrieve the public IP address of the new Ops Manager virtual machine.

2. Continue to the Configuring Ops Manager Director on Azure topic.

Deploying BOSH and Ops Manager to Azure Manually

Page last updated:

This topic describes how to deploy BOSH and Ops Manager for Pivotal Cloud Foundry (PCF) on Azure by using individual commands to create resources. Pivotal recommends this manual procedure for deploying to Azure China, Azure Germany, and Azure Government Cloud.

As an alternative to this procedure, you can use an Azure Resource Manager (ARM) template to create resources automatically. For information about using the ARM template, see the Deploying BOSH and Ops Manager to Azure with ARM topic.

Before you perform the procedures in this topic, you must have completed the procedures in the Preparing Azure topic. After you complete the procedures in this topic, follow the instructions in the Configuring Ops Manager on Azure topic.

Note: If you are deploying PCF on Azure Stack, complete the procedures in Install and configure CLI for use with Azure Stack in the Microsoft documentation before following the procedures in this topic.

Note: The Azure portal sometimes displays the names of resources with incorrect capitalization. Always use the Azure CLI to retrieve the correctly capitalized name of a resource.

Step 1: Create Network Resources

1. Navigate to the Azure portal, click **Resource groups**, and click **Add** to create a new resource group for your PCF deployment.

2. Enter a **Resource group name**, select your **Subscription**, and select a **Resource group location**. Click **Create**.

3. Export the name of your resource group as the environment variable `$RESOURCE_GROUP`.

```
$ export RESOURCE_GROUP="YOUR-RESOURCE-GROUP-NAME"
```

   Note: If you are on a Windows machine, you can use `set` instead of `export`.

4. Export your location. For example, `westus`.

```
$ export LOCATION=westus
```

   Note: For a list of available locations, run `az account list-locations`.

5. Create a network security group named `pcf-nsg`.

```
$ az network nsg create --name pcf-nsg \
--resource-group $RESOURCE_GROUP \
--location $LOCATION
```

6. Add network security group rules to the `pcf-nsg` group to allow traffic to known ports from the public Internet.

```
$ az network nsg rule create --name ssh \
--nsg-name pcf-nsg --resource-group $RESOURCE_GROUP \
--protocol Tcp --priority 100 \
--destination-port-range '22'
```

```
$ az network nsg rule create --name http \
--nsg-name pcf-nsg --resource-group $RESOURCE_GROUP \
--protocol Tcp --priority 200 \
--destination-port-range '80'
```

```
$ az network nsg rule create --name https \
--nsg-name pcf-nsg --resource-group $RESOURCE_GROUP \
--protocol Tcp --priority 300 \
--destination-port-range '443'
```

7. Create a network security group named `opsmgr-nsg`.

```
$ az network nsg create --name opsmgr-nsg \
--resource-group $RESOURCE_GROUP \
--location $LOCATION
```

8. Add a network security group rule to the `opsmgr-nsg` group to allow HTTP traffic to the Ops Manager VM.

```
$ az network nsg rule create --name http \
--nsg-name opsmgr-nsg --resource-group $RESOURCE_GROUP \
--protocol Tcp --priority 100 \
--destination-port-range 80
```

9. Add a network security group rule to the `opsmgr-nsg` group to allow HTTPS traffic to the Ops Manager VM.

```
$ az network nsg rule create --name https \
--nsg-name opsmgr-nsg --resource-group $RESOURCE_GROUP \
--protocol Tcp --priority 200 \
--destination-port-range 443
```

10. Add a network security group rule to the `opsmgr-nsg` group to allow SSH traffic to the Ops Manager VM.

```
$ az network nsg rule create --name ssh \
--nsg-name opsmgr-nsg --resource-group $RESOURCE_GROUP \
--protocol Tcp --priority 300 \
--destination-port-range 22
```

11. Create a virtual network named `pcf-net`.

```
$ az network vnet create --name pcf-net \
--resource-group $RESOURCE_GROUP --location $LOCATION \
--address-prefixes 10.0.0.0/16
```

12. Add a subnet to the network for PCF VMs.

```
$ az network vnet subnet create --name pcf \
--vnet-name pcf-net \
--resource-group $RESOURCE_GROUP \
--address-prefix 10.0.0.0/20
```

Step 2: Create BOSH and Deployment Storage Accounts

Azure for PCF uses multiple general-purpose Azure storage accounts. The BOSH and Ops Manager VMs use one main BOSH account, and the other components share five or more deployment storage accounts.

1. Choose a name for your BOSH storage account, and export it as the environment variable `$STORAGE_NAME`. Storage account names must be globally unique across Azure, between 3 and 24 characters in length, and contain only lowercase letters and numbers.

```
$ export STORAGE_NAME="YOUR-BOSH-STORAGE-ACCOUNT-NAME"
```

2. Create a Standard storage account for BOSH with the following command. This account will be used for BOSH bookkeeping and running the Ops Manager VM itself, but does not have to be used for running any other VMs.

```
$ az storage account create --name $STORAGE_NAME \
--resource-group $RESOURCE_GROUP \
--sku Standard_LRS \
--location $LOCATION
```

> **Note:** `Standard_LRS` refers to a Standard Azure storage account. The BOSH Director requires table storage to store stemcell information. Azure Premium storage does not support table storage and cannot be used for the BOSH storage account.

If the command fails, ensure you have followed the rules for naming your storage account. Export another new storage account name if necessary.

3. Configure the Azure CLI to use the BOSH storage account as its default.

a. Retrieve the connection string for the account.

```
$ az storage account show-connection-string \
--name $STORAGE_NAME --resource-group $RESOURCE_GROUP
```

The command returns output similar to the following:

```
{
"connectionString": "DefaultEndpointsProtocol=https;EndpointSuffix=core.windows.net;AccountName=cfdocsboshstorage;AccountKey=EXAMPLEaaaaabbbrnc5igFxYWsgq016
}
```

b. Record the full value of `connectionString` from the output above, starting with and including `DefaultEndpointsProtocol=`.
c. Export the value of `connectionString` as the environment variable `$CONNECTION_STRING`.

```
$ export CONNECTION_STRING="YOUR-ACCOUNT-KEY-STRING"
```

4. Create three blob containers in the BOSH storage account, named `opsmanager`, `bosh`, and `stemcell`.

```
$ az storage container create --name opsmanager \
--connection-string $CONNECTION_STRING
$ az storage container create --name bosh \
--connection-string $CONNECTION_STRING
$ az storage container create --name stemcell --public-access blob \
--connection-string $CONNECTION_STRING
```

5. Create a table named `stemcells`.

```
$ az storage table create --name stemcells \
--connection-string $CONNECTION_STRING
```

6. Choose a set of unique names for five or more deployment storage accounts. As with the BOSH storage account above, the names must be unique, alphanumeric, lowercase, and 3-24 characters long. The account names must also be sequential or otherwise identical except for the last character. For example: `xyzdeploystorage1`, `xyzdeploystorage2`, `xyzdeploystorage3`, `xyzdeploystorage4`, and `xyzdeploystorage5`.

7. Decide which type of storage to use and run the corresponding command below:

Note: Pivotal recommends five Premium storage accounts, which provides a reasonable amount of initial storage capacity. You can use either Premium or Standard storage accounts, but they have very different scalability metrics. Pivotal recommends creating 1 Standard storage account for every 30 VMs, or 1 Premium storage account for every 150 VMs. You can increase the number of storage accounts later by provisioning more and following the naming sequence.

- To use Premium storage (recommended):

```
$ export STORAGE_TYPE="Premium_LRS"
```

- To use Standard storage:

```
$ export STORAGE_TYPE="Standard_LRS"
```

8. For **each deployment storage account**, do the following:

a. Create the storage account with the following command, replacing `MY_DEPLOYMENT_STORAGE_X` with one of your deployment storage account names and `SUBSCRIPTION_ID` with your subscription ID.

```
$ az storage account create --name MY_DEPLOYMENT_STORAGE_X \
--resource-group $RESOURCE_GROUP --sku $STORAGE_TYPE \
--kind Storage --location $LOCATION
```

If the command fails, try a different set of account names.
b. Retrieve the connection string for the account.

```
$ az storage account show-connection-string \
--name MY_DEPLOYMENT_STORAGE_X --resource-group $RESOURCE_GROUP
```

The command returns output similar to the following:

```
{
"connectionString": "DefaultEndpointsProtocol=https;EndpointSuffix=core.windows.net;AccountName=cfdocsdeploystorage1;AccountKey=EXAMPLEaaaaaaaQiSAmqj1OocsGh
}
```

c. Record the full value of `connectionString` from the output above, starting with and including `DefaultEndpointsProtocol=`.
d. Create two blob containers named `bosh` and `stemcell` in the account.

```
$ az storage container create --name bosh \
--connection-string $CONNECTION_STRING
```

```
$ az storage container create --name stemcell \
--connection-string $CONNECTION_STRING
```

Step 3: Create a Load Balancer

1. Create a load balancer named `pcf-lb`.

```
$ az network lb create --name pcf-lb \
--resource-group $RESOURCE_GROUP --location $LOCATION
```

2. Create a static IP address for the load balancer named `pcf-lb-ip`.

```
$ az network public-ip create --name pcf-lb-ip \
--resource-group $RESOURCE_GROUP --location $LOCATION \
--allocation-method Static
{
  "publicIp": {
    "dnsSettings": null,
    "etag": "W/\"a9b780e8-38bc-4a28-a563-e13a5859169d\"",
    "id": "/subscriptions/995b7eed-77ef-45ff-a5c9-1a405ffb8243/resourceGroups/cf-docs/providers/Microsoft.Network/publicIPAddresses/pcf-lb-ip",
    "idleTimeoutInMinutes": 4,
    "ipAddress": "13.64.255.40",
    "ipConfiguration": null,
    "location": "westus",
    "name": "pcf-lb-ip",
    "provisioningState": "Succeeded",
    "publicIpAddressVersion": "IPv4",
    "publicIpAllocationMethod": "Static",
    "resourceGroup": "cf-docs",
    "resourceGuid": "4fbf2fe5-6f7e-449a-ae70-e513a9f5cddc",
    "tags": null,
    "type": "Microsoft.Network/publicIPAddresses"
  }
}
```

3. Record the `ipAddress` from the output above. This is the public IP address of your load balancer.

4. Add a front-end IP configuration to the load balancer.

```
$ az network lb frontend-ip create --lb-name pcf-lb \
--name pcf-fe-ip --resource-group $RESOURCE_GROUP \
--public-ip-address pcf-lb-ip
```

5. Add a probe to the load balancer.

```
$ az network lb probe create --lb-name pcf-lb \
--name tcp80 --resource-group $RESOURCE_GROUP \
--protocol Tcp --port 80
```

6. Add a backend address pool to the load balancer.

```
$ az network lb address-pool create --lb-name pcf-lb \
--name pcf-lbbepool --resource-group $RESOURCE_GROUP
```

Note: This backend pool is empty when you create it.

7. Add a load balancing rule for HTTP.

```
$ az network lb rule create --lb-name pcf-lb \
--name http --resource-group $RESOURCE_GROUP \
--protocol Tcp --frontend-port 80 \
--backend-port 80 --frontend-ip-name pcf-fe-ip \
--backend-pool-name pcf-lbbepool \
--probe-name tcp80
```

8. Add a load balancing rule for HTTPS.

```
$ az network lb rule create --lb-name pcf-lb \
--name https --resource-group $RESOURCE_GROUP \
--protocol Tcp --frontend-port 443 \
--backend-port 443 --frontend-ip-name pcf-fe-ip \
--backend-pool-name pcf-lbbepool \
--probe-name tcp80
```

9. Add a load balancing rule for SSH.

```
$ az network lb rule create --lb-name pcf-lb \
--name diego-ssh --resource-group $RESOURCE_GROUP \
--protocol Tcp --frontend-port 2222 \
--backend-port 2222 --frontend-ip-name pcf-fe-ip \
--backend-pool-name pcf-lbbepool
--probe-name tcp80
```

10. Navigate to your DNS provider, and create an entry that points `*.YOUR-SUBDOMAIN` to the public IP address of your load balancer that you recorded in a previous step. For example, create an entry that points `azure.example.com` to `198.51.100.1`.

> **Note**: If you did not record the IP address of your load balancer earlier, you can retrieve it by navigating to the Azure portal, clicking **All resources**, and clicking the **Public IP address** resource that ends with `pcf-lb-ip`.

Step 4: Boot Ops Manager

1. Navigate to Pivotal Network and download the latest release of **Pivotal Cloud Foundry Ops Manager for Azure**.
2. View the downloaded PDF and locate the Ops Manager image URL appropriate for your region.
3. Export the Ops Manager image URL as an environment variable.

```
$ export OPS_MAN_IMAGE_URL="YOUR-OPS-MAN-IMAGE-URL"
```

4. Download the Ops Manager image. For compatibility when upgrading to future versions of Ops Manager, choose a unique name for the image that includes the Ops Manager version number. For example, replace `opsman-image-2.0.x` in the following examples with `opsman-image-2.0.1`.
   - If you use unmanaged disks, perform the following steps:

     > **Note**: Azure Stack requires unmanaged disks.

     1. Download the Ops Manager image to your local machine. The image size is 5 GB.

     ```
     $ wget $OPS_MAN_IMAGE_URL -O opsman-image-2.0.x.vhd
     ```

     2. Upload the image to your storage account using the Azure CLI.

     ```
     $ az storage blob upload --name opsman-image-2.0.x.vhd \
     --connection-string $CONNECTION_STRING \
     --container-name opsmanager \
     --type page \
     --file opsman-image-2.0.x.vhd
     ```

   - If you use managed disks, perform the following steps:

     1. Copy the Ops Manager image into your storage account using the Azure CLI.

```
$ az storage blob copy start --source-uri $OPS_MAN_IMAGE_URL \
--connection-string $CONNECTION_STRING \
--destination-container opsmanager \
--destination-blob opsman-image-2.0.x.vhd
```

2. Copying the image may take several minutes. Run the following command and examine the output under `"copy"`:

```
$ az storage blob show --name opsman-image-2.0.x.vhd \
--container-name opsmanager \
--account-name $STORAGE_NAME
...
"copy": {
"completionTime": "2017-06-26T22:24:11+00:00",
"id": "b9c8b272-a562-4574-baa6-f1a04afcefdf",
"progress": "53687091712/53687091712",
"source": "https://opsmanagerwestus.blob.core.windows.net/images/ops-manager-2.0.0.vhd",
"status": "success",
"statusDescription": null
},
```

3. Wait a few moments and re-run the command above if `status` is `pending`. When `status` reads `success`, continue to the next step.

5. Create a public IP address named `ops-manager-ip`.

```
$ az network public-ip create --name ops-manager-ip \
--resource-group $RESOURCE_GROUP --location $LOCATION \
--allocation-method Static
{
  "publicIp": {
    "dnsSettings": null,
    "etag": "W/\"4450ebe2-9e97-4b17-9cf2-44838339c661\"",
    "id": "/subscriptions/995b7eed-77ef-45ff-a5c9-1a405ffb8243/resourceGroups/cf-docs/providers/Microsoft.Network/publicIPAddresses/ops-manager-ip",
    "idleTimeoutInMinutes": 4,
    "ipAddress": "40.83.148.183",
    "ipConfiguration": null,
    "location": "westus",
    "name": "ops-manager-ip",
    "provisioningState": "Succeeded",
    "publicIpAddressVersion": "IPv4",
    "publicIpAllocationMethod": "Static",
    "resourceGroup": "cf-docs",
    "resourceGuid": "950d4831-1bec-42da-8a79-959bcddea9dd",
    "tags": null,
    "type": "Microsoft.Network/publicIPAddresses"
  }
}
```

6. Record the `ipAddress` from the output above. This is the public IP address of Ops Manager.

7. Create a network interface for Ops Manager.

```
$ az network nic create --vnet-name pcf-net \
--subnet pcf --network-security-group opsmgr-nsg \
--private-ip-address 10.0.0.5 \
--public-ip-address ops-manager-ip \
--resource-group $RESOURCE_GROUP \
--name opsman-nic --location $LOCATION
```

8. Create a keypair on your local machine with the username `ubuntu`. For example, enter the following command:

```
$ ssh-keygen -t rsa -f opsman -C ubuntu
```

When prompted for a passphrase, press the `enter` key to provide an empty passphrase.

9. Create the Ops Manager VM.
   - If you are using unmanaged disks, run the following command to create your Ops Manager VM, replacing `PATH-TO-PUBLIC-KEY` with the path to your public key .pub file:

```
$ az vm create --name opsman-2.0.x --resource-group $RESOURCE_GROUP \
--location $LOCATION \
--nics opsman-nic \
--image https://$STORAGENAME.my-azure-instance.com/opsmanager/opsman-image-2.0.x.vhd \
--os-disk-name opsman-2.0.x-osdisk \
--os-disk-size-gb 128 \
--os-type Linux \
--use-unmanaged-disk \
--storage-account $STORAGENAME \
--storage-container-name opsmanager \
--admin-username ubuntu \
--ssh-key-value PATH-TO-PUBLIC-KEY
```

Replace `my-azure-instance.com` with the URL of your Azure instance. Find the complete source URL in the Azure UI by viewing the **Blob properties** of the Ops Manager image you created earlier in this procedure.

- If you are using Azure managed disks, perform the following steps:

1. Create a managed image from the Ops Manager VHD file:

```
$ az image create --resource-group $RESOURCE_GROUP \
--name opsman-image-2.0.x \
--source https://$STORAGE_NAME.blob.core.windows.net/opsmanager/image-2.0.x.vhd \
--location $LOCATION \
--os-type Linux
```

If you are using Azure China, Azure Government Cloud, or Azure Germany, replace `blob.core.windows.net` with the following:

- For Azure China, use `blob.core.chinacloudapi.cn`. See the Azure documentation for more information.
- For Azure Government Cloud, use `blob.core.usgovcloudapi.net`. See the Azure documentation for more information.
- For Azure Germany, use `blob.core.cloudapi.de`. See the Azure documentation for more information.

2. Create your Ops Manager VM, replacing `PATH-TO-PUBLIC-KEY` with the path to your public key `.pub` file.

```
$ az vm create --name opsman-2.0.x --resource-group $RESOURCE_GROUP \
--location $LOCATION \
--nics opsman-nic \
--image opsman-image-2.0.x \
--os-disk-name opsman-2.0.x-osdisk \
--admin-username ubuntu \
--size Standard_DS2_v2 \
--storage-sku Standard_LRS \
--ssh-key-value PATH-TO-PUBLIC-KEY
```

10. If you plan to install more than one tile in this Ops Manager installation, perform the following steps to increase the size of the Ops Manager VM disk. You can repeat this process and increase the disk again at a later time if necessary.

Note: If you use Azure Stack, you must increase the Ops Manager VM disk size using the Azure Stack UI.

a. Run the following command to stop the VM and detach the disk:

```
$ az vm deallocate --name opsman-2.0.x \
--resource-group $RESOURCE_GROUP
```

b. Run the following command to resize the disk to 128 GB:

```
$ az disk update --size-gb 128 --name opsman-2.0.x-osdisk \
--resource-group $RESOURCE_GROUP
```

c. Run the following command to start the VM:

```
$ az vm start --name opsman-2.0.x --resource-group $RESOURCE_GROUP
```

Step 5: Complete Ops Manager Director Configuration

1. Navigate to your DNS provider, and create an entry that points a fully qualified domain name (FQDN) to the public IP address of Ops Manager. As a best practice, always use the FQDN to access Ops Manager.

2. Continue to the Configuring Ops Manager on Azure topic.

Configuring Ops Manager on Azure

Page last updated:

This topic describes how to configure the Ops Manager Director for Pivotal Cloud Foundry (PCF) on Azure.

Prerequisites

Before you perform the procedures in this topic, you must have completed the procedures in the following topics:

- The Preparing to Deploy PCF on Azure topic
- Either the Launching an Ops Manager Director Instance with an ARM Template topic or the Launching an Ops Manager Director Instance on Azure without an ARM Template topic

Note: You can also perform the procedures in this topic using the Ops Manager API. For more information, see the Using the Ops Manager API topic.

Prerequisites for Azure Stack

Azure Stack uses a different set of VM types than a standard Azure deployment. To install the Ops Manager Director tile on Azure Stack, you must set your virtual machine (VM) types using the Ops Manager API.

Setting VM types with the API modifies the VMs available in the **Resource Config** pane of the tile. For a current list of Azure Stack VM types that you can include in your API call, see Virtual Machine Sizes from the Azure Stack documentation. Ensure that you provide at least one `Standard` VM type for use by the **Router** VM of the Pivotal Application Service (PAS) tile. Below is an example command:

```
$ curl -k https://YOUR-OPS-MAN-FQDN/api/v0/vm_types -X
PUT -H "Authorization: bearer YOUR-ACCESS-TOKEN" -H
"Content-Type: application/json" -d '{"vm_types":[
{"name":"Standard_DS1_v2","ram":3584,"cpu":1,"ephemeral_disk":51200},
{"name":"Standard_DS2_v2","ram":7168,"cpu":2,"ephemeral_disk":102400},
{"name":"Standard_DS3_v2","ram":14336,"cpu":4,"ephemeral_disk":204800},
{"name":"Standard_DS4_v2","ram":28672,"cpu":8,"ephemeral_disk":409600},
{"name":"Standard_DS5_v2","ram":57344,"cpu":8,"ephemeral_disk":819200},
{"name":"Standard_DS11_v2","ram":14336,"cpu":2,"ephemeral_disk":102400},
{"name":"Standard_DS12_v2","ram":28672,"cpu":4,"ephemeral_disk":204800},
{"name":"Standard_DS13_v2","ram":57344,"cpu":8,"ephemeral_disk":409600},
{"name":"Standard_DS14_v2","ram":114688,"cpu":16,"ephemeral_disk":819200}]}'
```

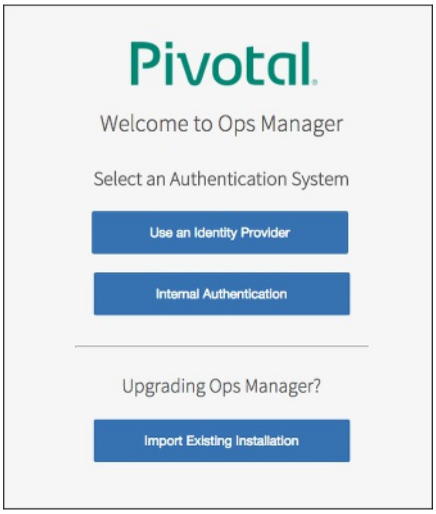
Note: To obtain your authorization token, perform the procedures in the Using the Ops Manager API topic.

WARNING: If you do not set your VM types, the Ops Manager deploy will fail. You must set your VM types before configuring resources and clicking **Apply Changes**.

Step 1: Access Ops Manager

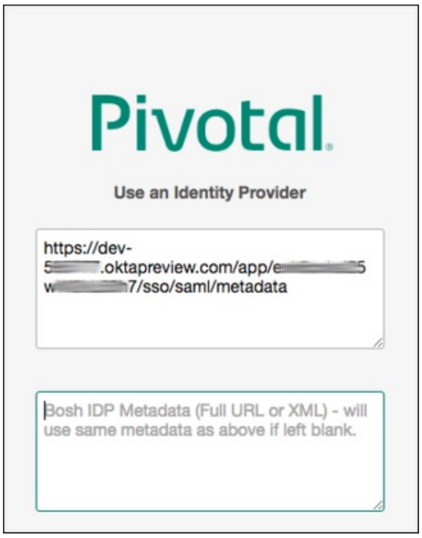
1. In a web browser, navigate to the fully qualified domain name (FQDN) of Ops Manager that you set up in either the Launching an Ops Manager Director Instance with an ARM Template topic or the Launching an Ops Manager Director Instance on Azure without an ARM Template topic.

2. When Ops Manager starts for the first time, you must choose one of the following:
   - Use an Identity Provider: If you use an Identity Provider, an external identity server maintains your user database.
   - Internal Authentication: If you use Internal Authentication, PCF maintains your user database.



Use an Identity Provider (IdP)

1. Log in to your IdP console and download the IdP metadata XML. Optionally, if your IdP supports metadata URL, you can copy the metadata URL instead of the XML.



2. Copy the IdP metadata XML or URL to the Ops Manager **Use an Identity Provider** log in page.

> **Note**: The same IdP metadata URL or XML is applied for the BOSH Director. If you use a separate IdP for BOSH, copy the metadata XML or URL from that IdP and enter it into the BOSH IdP Metadata text box in the Ops Manager log in page.

3. Enter your **Decryption passphrase**. Read the **End User License Agreement**, and select the checkbox to accept the terms.

4. Your Ops Manager log in page appears. Enter your username and password. Click **Login**.

5. Download your SAML Service Provider metadata (SAML Relying Party metadata) by navigating to the following URLs:
   - **5a.** Ops Manager SAML service provider metadata: `https://OPS-MAN-FQDN:443/uaa/saml/metadata`
   - **5b.** BOSH Director SAML service provider metadata: `https://BOSH-IP-ADDRESS:8443/saml/metadata`

> **Note**: To retrieve your `BOSH-IP-ADDRESS`, navigate to the **Ops Manager Director** tile > **Status** tab. Record the **Ops Manager Director** IP address.

6. Configure your IdP with your SAML Service Provider metadata. Import the Ops Manager SAML provider metadata from Step 5a above to your IdP. If your IdP does not support importing, provide the values below.
   - **Single sign on** URL: `https://OPS-MAN-FQDN:443/uaa/saml/SSO/alias/OPS-MAN-FQDN`
   - **Audience URI** (SP Entity ID): `https://OP-MAN-FQDN:443/uaa`
   - **Name ID**: Email Address

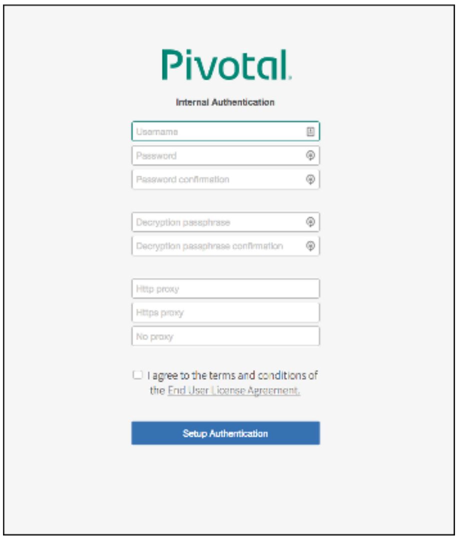
- SAML authentication requests are always signed

7. Import the BOSH Director SAML provider metadata from Step 5b to your IdP. If the IdP does not support an import, provide the values below.
   - **Single sign on** URL: `https://BOSH-IP:8443/saml/SSO/alias/BOSH-IP`
   - **Audience URI** (SP Entity ID): `https://BOSH-IP:8443`
   - **Name ID**: Email Address
   - SAML authentication requests are always signed

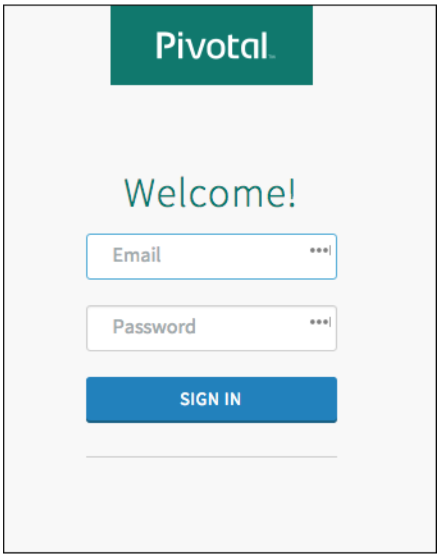
8. Return to the **Ops Manager Director** tile, and continue with the configuration steps below.

Internal Authentication

1. When redirected to the **Internal Authentication** page, you must complete the following steps:
   - Enter a **Username**, **Password**, and **Password confirmation** to create an Admin user.
   - Enter a **Decryption passphrase** and the **Decryption passphrase confirmation**. This passphrase encrypts the Ops Manager datastore, and is not recoverable if lost.
   - If you are using an **HTTP proxy** or **HTTPS proxy**, follow the instructions in the Configuring Proxy Settings for the BOSH CPI topic.
   - Read the **End User License Agreement**, and select the checkbox to accept the terms.
   - Click **Setup Authentication**.



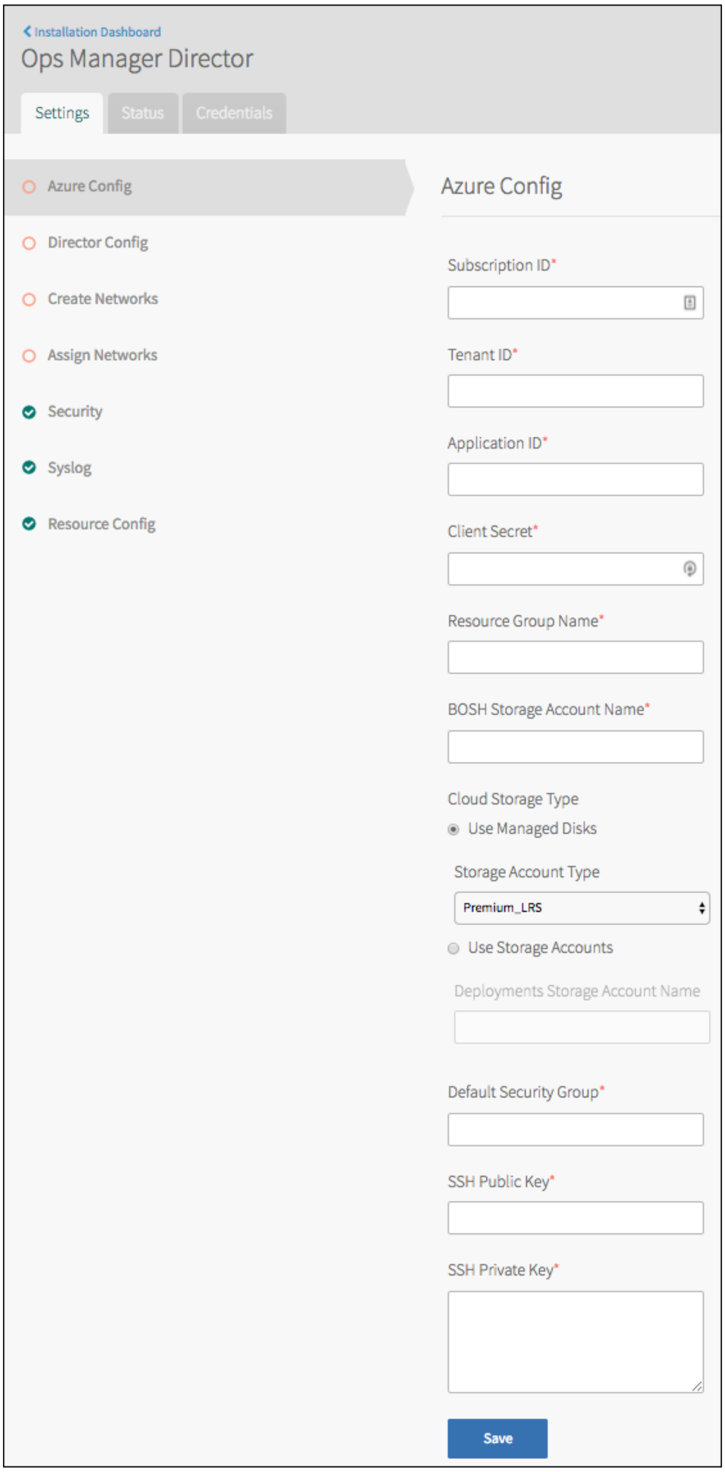
2. Log in to Ops Manager with the Admin username and password that you created in the previous step.



Step 2: Azure Config Page

1. Click the **Ops Manager Director** tile.

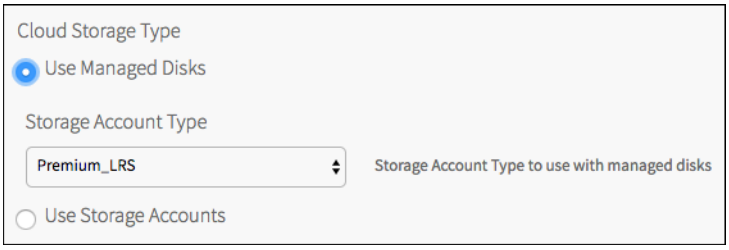
2. Select **Azure Config**.



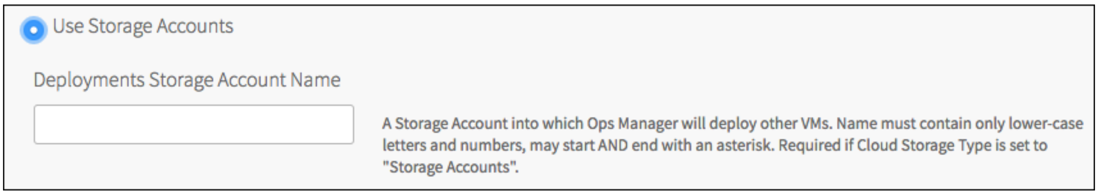
3. Complete the following fields with information you obtained in the Preparing to Deploy PCF on Azure topic.
   - **Subscription ID**: Enter the ID of your Azure subscription.
   - **Tenant ID**: Enter your `TENANT_ID`.
   - **Application ID**: Enter the `APPLICATION_ID` that you created in the Create an Azure Active Directory Application step of the *Preparing to Deploy PCF on Azure* topic.
   - **Client Secret**: Enter your `CLIENT_SECRET`.

4. Complete the following fields:
   - **Resource Group Name**: Enter the name of your resource group, which you exported as the `$RESOURCE_GROUP` environment variable.
   - **BOSH Storage Account Name**: Enter the name of your storage account, which you exported as the `$STORAGE_NAME` environment variable.

5. For **Cloud Storage Type**, select one of the following options based on your Azure VM storage settings.
   - **Use Managed Disks**: Select this option if you use Azure Managed Disks. See Azure Managed Disks Overview in the Microsoft documentation for more information. For **Storage Account Type**, select the storage option that corresponds with your Azure subscription. Select **Standard_LRS** for HDD-based storage or **Premium_LRS** for SSD-based storage.



   - **Use Storage Accounts**: Select this option if you use storage accounts to store your Azure VMs. Enter the base storage name that you used to create your deployment storage accounts, prepended and appended with the wildcard character `*`. For example, if you created accounts named `xyzdeploymentstorage1`, `xyzdeploymentstorage2`, and `xyzdeploymentstorage3`, enter `*deploymentstorage*`. Ops Manager requires that you specify an asterisk at both the beginning and the end of the base storage account name.



6. For **Default Security Group**, enter `pcf-nsg`.

   > **Note**: The Azure portal sometimes displays the names of resources with incorrect capitalization. Always use the Azure CLI to retrieve the correctly capitalized name of a resource.

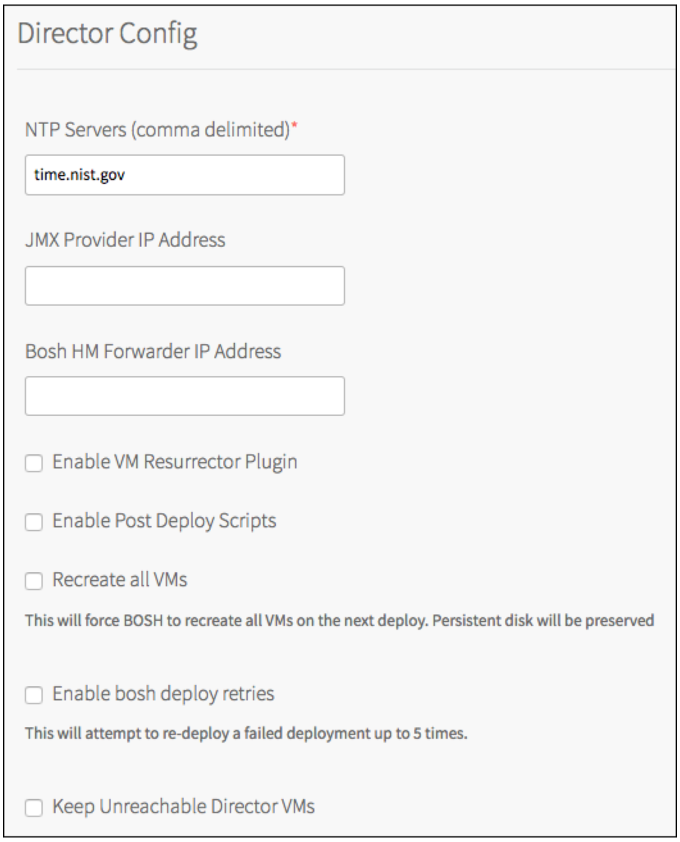
7. For **SSH Public Key**, copy and paste the contents of your public key in the `opsman.pub` file. You created this file in either the Launching an Ops Manager Director Instance with an ARM Template topic or the Launching an Ops Manager Director Instance on Azure without an ARM Template topic.

8. For **SSH Private Key**, copy and paste the contents of your private key in the `opsman` file.

9. Click **Save**.

Step 3: Director Config Page

1. Select **Director Config**.



2. In the **NTP Servers (comma delimited)** field, enter a comma-separated list of valid NTP servers.

3. Leave the **JMX Provider IP Address** field blank.

   > **Note**: Starting from PCF v2.0, BOSH-reported system metrics are available in the Loggregator Firehose by default. Therefore, if you continue to use PCF JMX Bridge for consuming them outside of the Firehose, you may receive duplicate data. To prevent this, leave the **JMX Provider IP Address** field blank. For additional guidance, see BOSH System Metrics Available in Loggregator Firehose.

4. Leave the **Bosh HM Forwarder IP Address** field blank.

   > **Note**: Starting from PCF v2.0, BOSH-reported system metrics are available in the Loggregator Firehose by default. Therefore, if you continue to use the BOSH HM Forwarder for consuming them, you may receive duplicate data. To prevent this, leave the **Bosh HM Forwarder IP Address** field blank. For additional guidance, see BOSH System Metrics Available in Loggregator Firehose.

5. Select the **Enable VM Resurrector Plugin** checkbox to enable the Ops Manager Resurrector functionality and increase PAS availability.

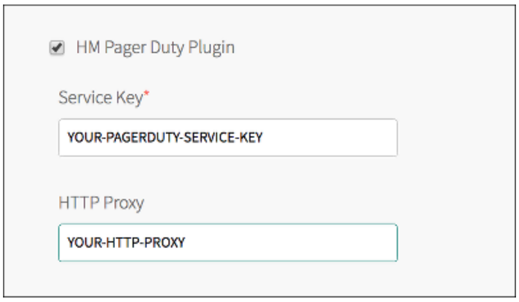
6. Select **Enable Post Deploy Scripts** to run a post-deploy script after deployment. This script allows the job to execute additional commands against a deployment.

7. Select **Recreate all VMs** to force BOSH to recreate all VMs on the next deploy. This process does not destroy any persistent disk data.

8. Select **Enable bosh deploy retries** if you want Ops Manager to retry failed BOSH operations up to five times.

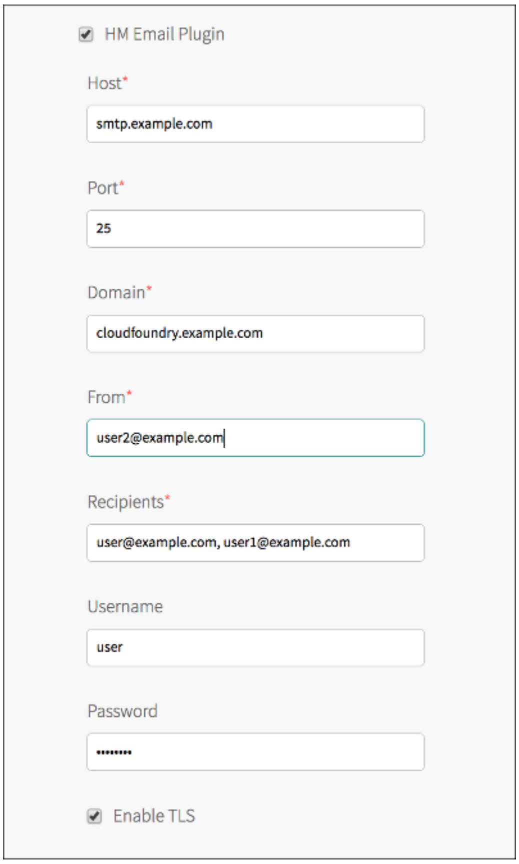
9. Select **Keep Unreachable Director VMs** if you want to preserve Ops Manager Director VMs after a failed deployment for troubleshooting purposes.

10. (Optional) Select **HM Pager Duty Plugin** to enable Health Monitor integration with PagerDuty.



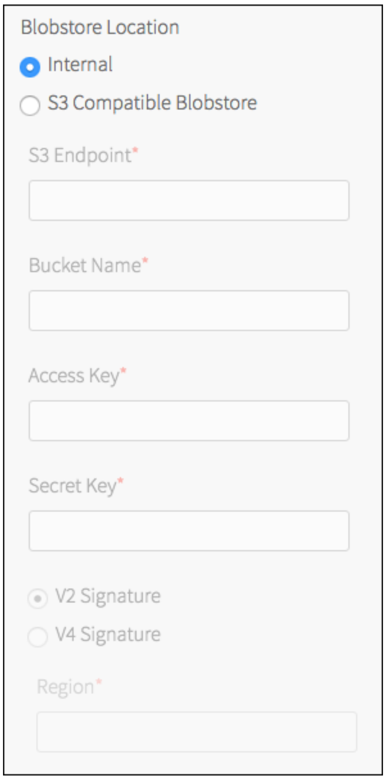
- **Service Key**: Enter your API service key from PagerDuty.
- **HTTP Proxy**: Enter an HTTP proxy for use with PagerDuty.

11. (Optional) Select **HM Email Plugin** to enable Health Monitor integration with email.

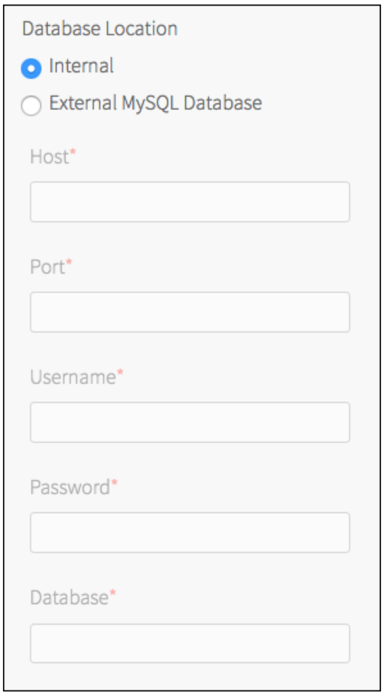


- **Host**: Enter your email hostname.
- **Port**: Enter your email port number.
- **Domain**: Enter your domain.
- **From**: Enter the address for the sender.
- **Recipients**: Enter comma-separated addresses of intended recipients.
- **Username**: Enter the username for your email server.
- **Password**: Enter the password password for your email server.
- **Enable TLS**: Select this checkbox to enable Transport Layer Security.

12. For **Blobstore Location**, Pivotal recommends that you keep **Internal** selected.



13. For **Database Location**, Pivotal recommends that you keep **Internal** selected.



14. (Optional) **Director Workers** sets the number of workers available to execute Director tasks. This field defaults to `5`.

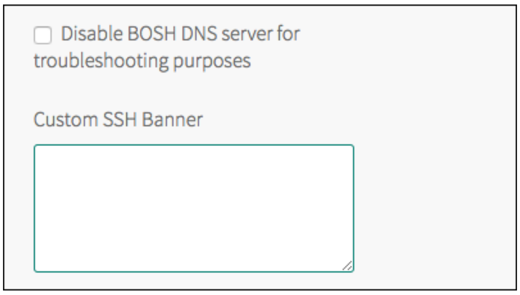
15. (Optional) **Max Threads** sets the maximum number of threads that the Ops Manager Director can run simultaneously. Pivotal recommends that you leave the field blank to use the default value unless doing so results in rate limiting or errors on your IaaS.

16. (Optional) To add a custom URL for your Ops Manager Director, enter a valid hostname in **Director Hostname**. You can also use this field to configure [a load balancer in front of your Ops Manager Director](#).

17. (Optional) To disable BOSH DNS, select the **Disable BOSH DNS server for troubleshooting purposes** checkbox. For more information about the BOSH DNS service discovery mechanism, see BOSH DNS Service Discovery for Application Containers in the Pivotal Application Service (PAS) Release Notes.

   Breaking Change: Do not disable BOSH DNS without consulting Pivotal Support. For more information about disabling BOSH DNS, see Disabling or Opting Out of BOSH DNS in PCF in the Pivotal Knowledge Base.

18. (Optional) To set a custom banner that users see when logging in to the Director using SSH, enter text in the **Custom SSH Banner** field.



19. Click **Save**.

Step 4: Create Networks Page

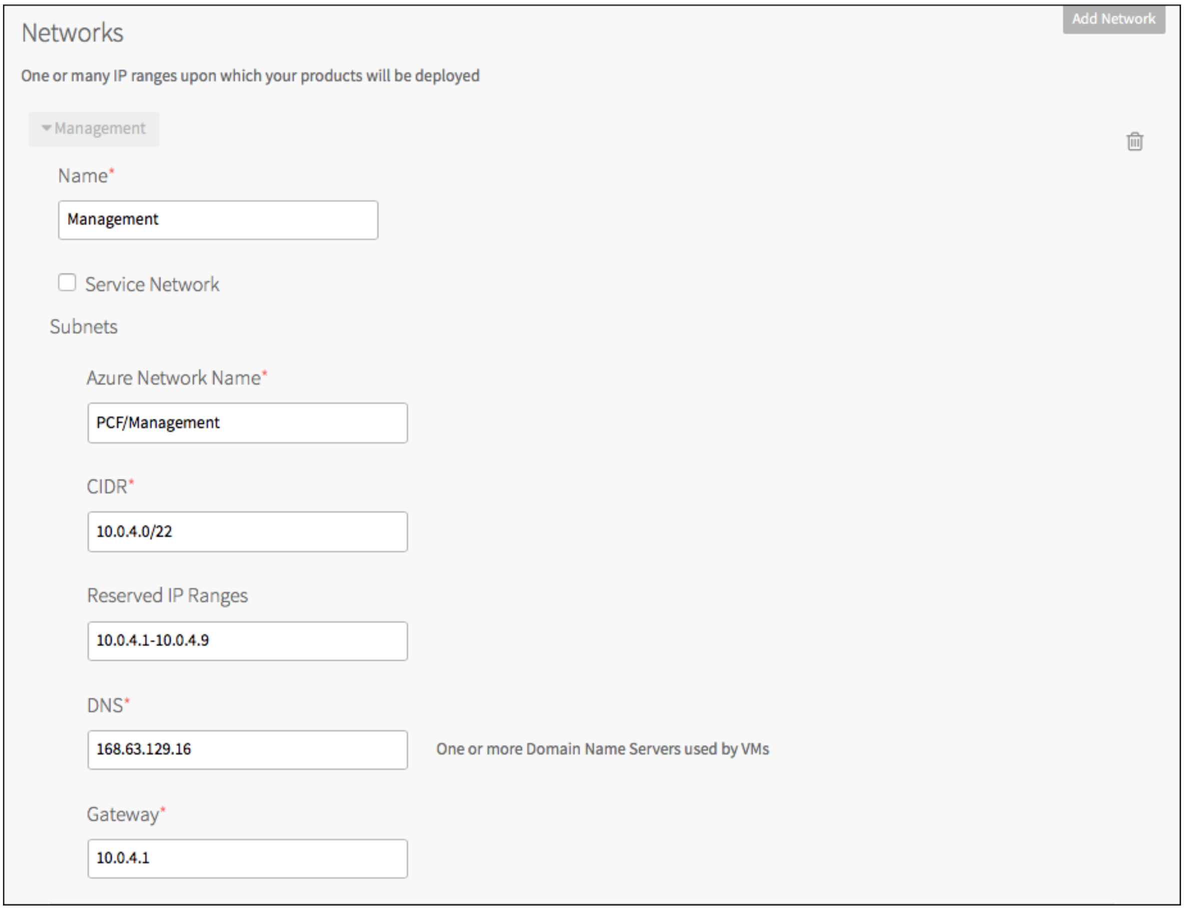
In this procedure, you create three networks.

Create the Management or Infrastructure Network

1. Select **Create Networks**.
2. Click **Add Network**.
3. Select **Enable ICMP checks** if you want to enable ICMP on your networks. Ops Manager uses ICMP checks to confirm that components within your network are reachable.
4. For **Name**, enter the name of the network you want to create. For example, `Management`.
5. Leave the **Service Network** checkbox deselected.
6. Under **Subnets**, complete the following fields:
   - **Azure Network Name**: Enter `PCF/Management`. You can use either the `NETWORK-NAME/SUBNET-NAME` format or the `RESOURCE-GROUP/NETWORK-NAME/SUBNET-NAME` format. If you specify a resource group, it must exist under the same subscription ID you provided in the Azure Config page.

     Note: The Azure portal sometimes displays the names of resources with incorrect capitalization. Always use the Azure CLI to retrieve the correctly capitalized name of a resource.

- **CIDR**: Enter `10.0.4.0/22`.
- **Reserved IP Ranges**: Enter the first 9 IP addresses of the subnet. For example, `10.0.4.1-10.0.4.9`.
- **DNS**: Enter `168.63.129.16`.
- **Gateway**: Enter the first IP address of the subnet. For example, `10.0.4.1`.



7. Click **Save**.

Create the Deployment or PAS Network

1. Click **Add Network**.

2. Select **Enable ICMP checks** if you want to enable ICMP on your networks. Ops Manager uses ICMP checks to confirm that components within your network are reachable.

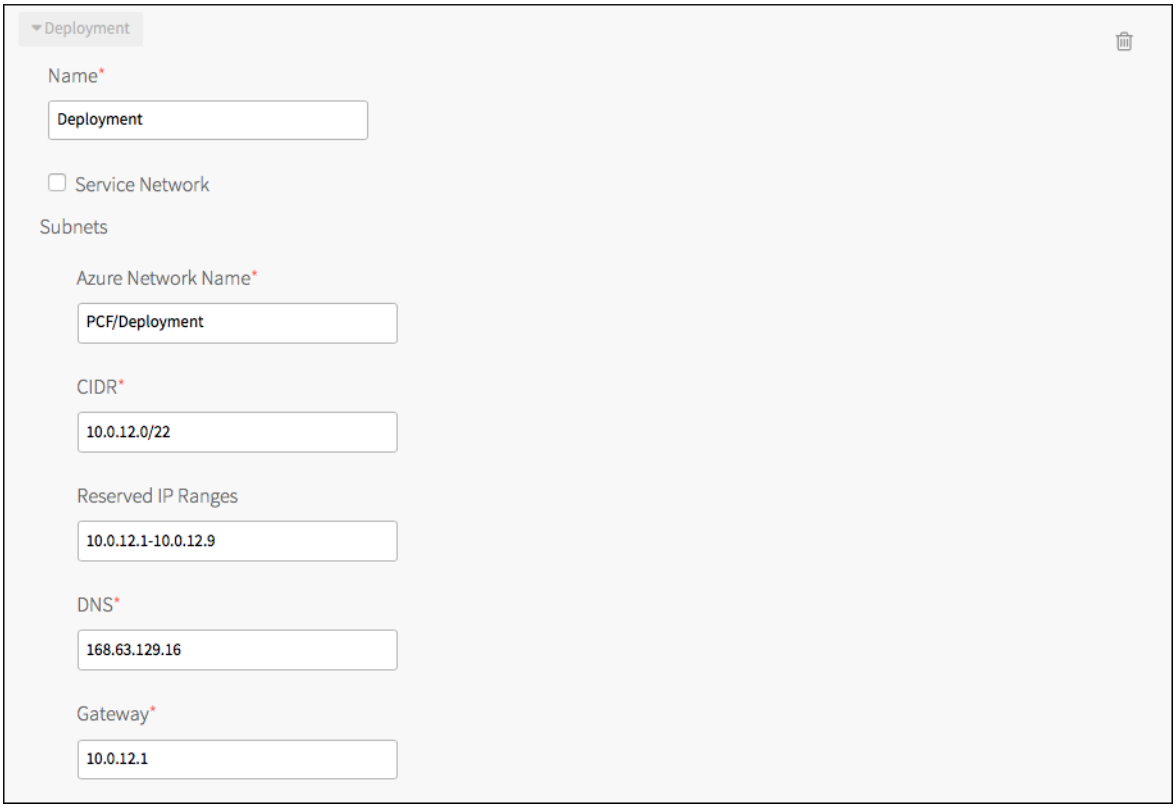
3. For **Name**, enter the name of the network you want to create. For example, `Deployment`.

4. Leave the **Service Network** checkbox deselected.

5. Under **Subnets**, complete the following fields:
   - **Azure Network Name**: Enter `PCF/Deployment`. You can use either the `NETWORK-NAME/SUBNET-NAME` format or the `RESOURCE-GROUP/NETWORK-NAME/SUBNET-NAME` format. If you specify a resource group, it must exist under the same subscription ID you provided in the Azure Config page.

     > **Note**: The Azure portal sometimes displays the names of resources with incorrect capitalization. Always use the Azure CLI to retrieve the correctly capitalized name of a resource.

   - **CIDR**: Enter `10.0.12.0/22`.
   - **Reserved IP Ranges**: Enter the first 9 IP addresses of the subnet. For example, `10.0.12.1-10.0.12.9`.
   - **DNS**: Enter `168.63.129.16`.
   - **Gateway**: Enter the first IP address of the subnet. For example, `10.0.12.1`.



6. Click **Save**.

Create the Services Network

1. Click **Add Network**.

2. Select **Enable ICMP checks** if you want to enable ICMP on your networks. Ops Manager uses ICMP checks to confirm that components within your network are reachable.

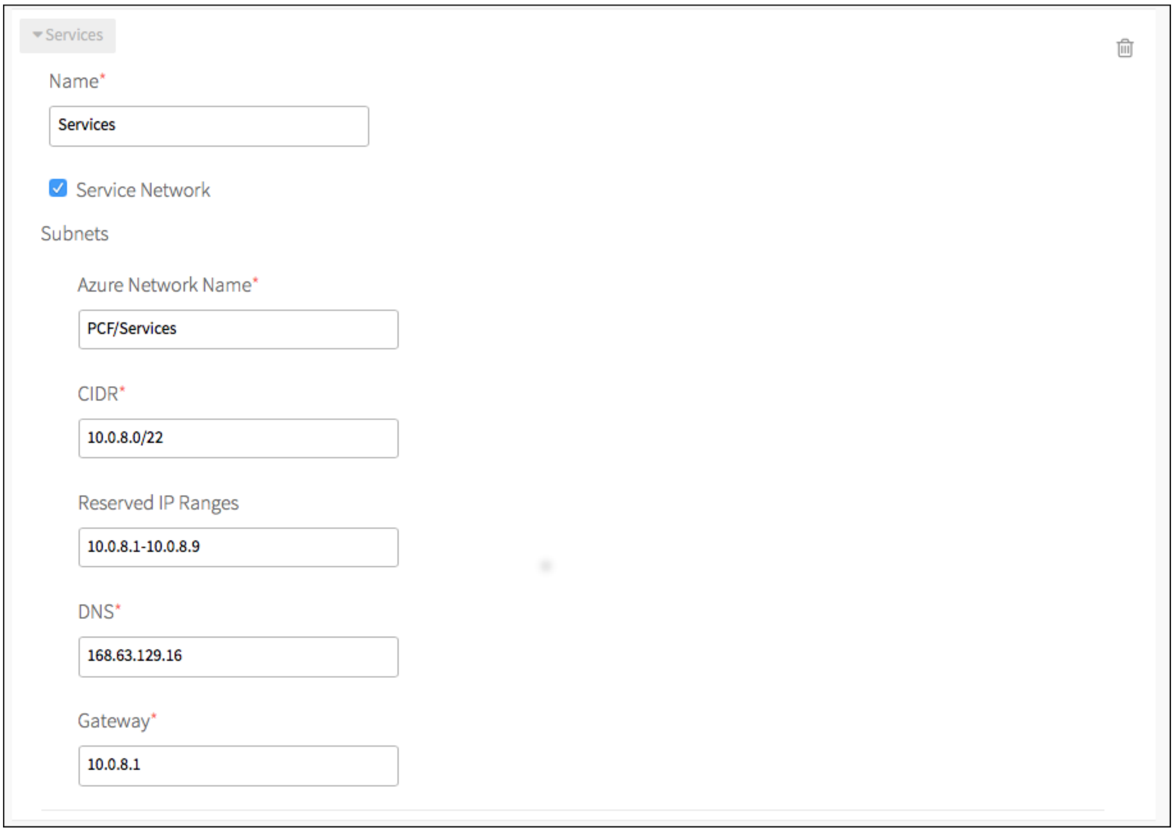
3. For **Name**, enter the name of the network you want to create. For example, `Services`.

4. If you want to dynamically provision VMs in this network for use with on-demand services, select the **Service Networks** checkbox. When the checkbox is selected, Ops Manager does not provision VMs within the specified CIDR range.

5. Under **Subnets**, complete the following fields:
    - **Azure Network Name**: Enter `PCF/Services`. You can use either the `NETWORK-NAME/SUBNET-NAME` format or the `RESOURCE-GROUP/NETWORK-NAME/SUBNET-NAME` format. If you specify a resource group, it must exist under the same subscription ID you provided in the Azure Config page.

      > **Note**: The Azure portal sometimes displays the names of resources with incorrect capitalization. Always use the Azure CLI to retrieve the correctly capitalized name of a resource.

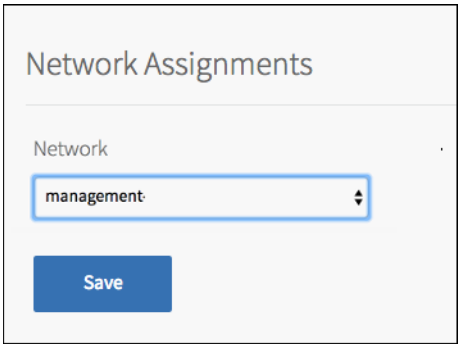
    - **CIDR**: Enter `10.0.8.0/22`.
    - **Reserved IP Ranges**: Enter the first 9 IP addresses of the subnet. For example, `10.0.8.1-10.0.8.9`.
    - **DNS**: Enter `168.63.129.16`.
    - **Gateway**: Enter the first IP address of the subnet. For example, `10.0.8.1`.



6. Click **Save**. If you do not have **Enable ICMP checks** selected, you may see red warnings which you can safely ignore.

Step 5: Assign Networks Page

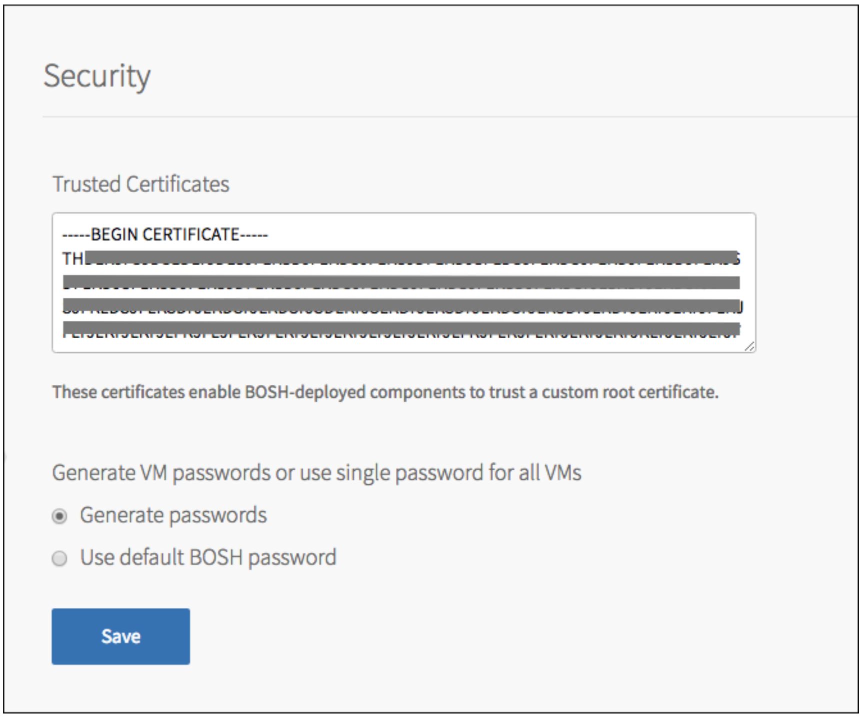
1. Select **Assign Networks**.



2. Under **Network**, select the `Management` network you created from the dropdown menu.
3. Click **Save**.

Step 6: Security Page

1. Select **Security**.



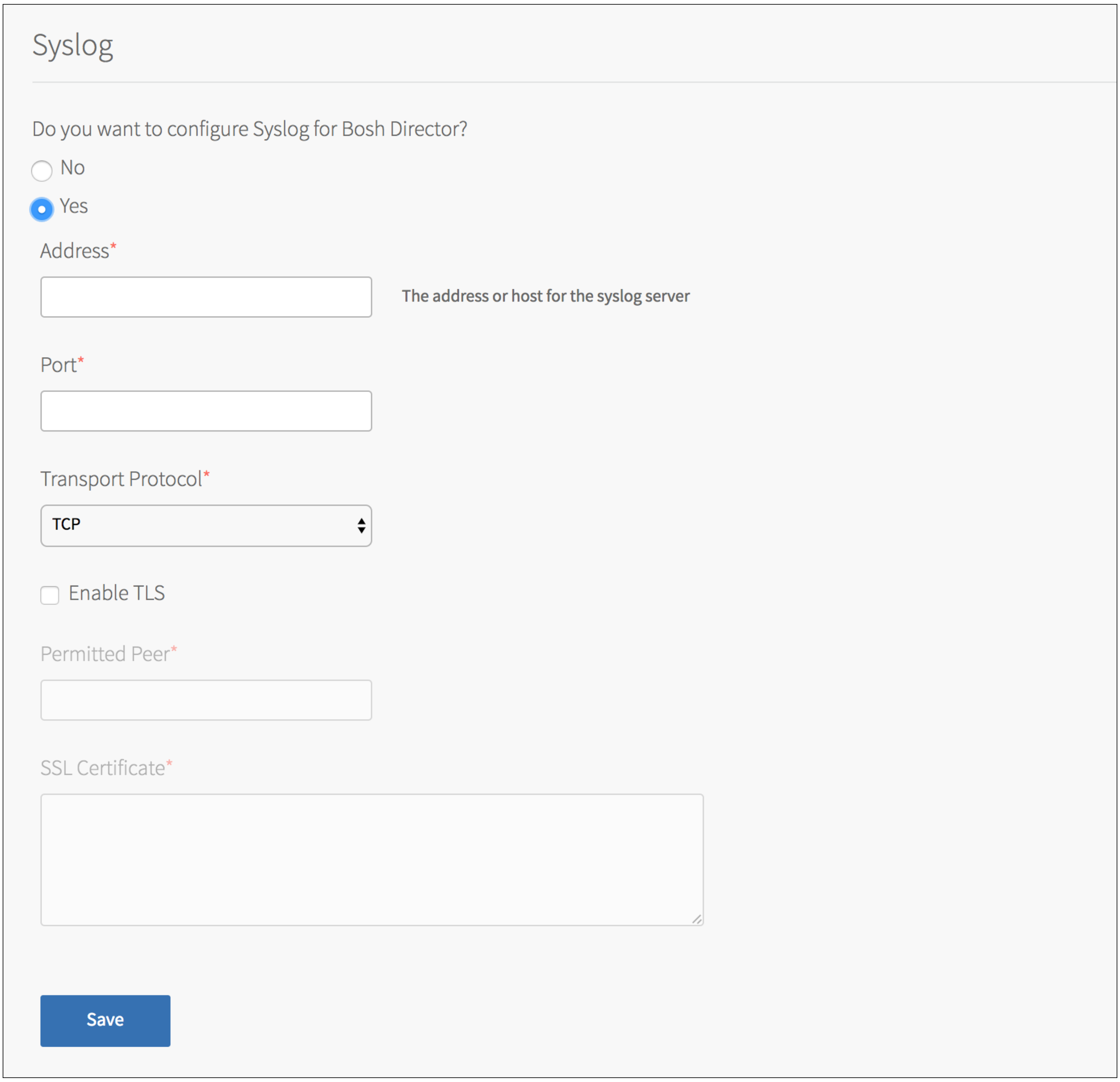
2. In **Trusted Certificates**, enter a custom Certificate Authority (CA) certificate to insert into your organization's certificate trust chain. This feature enables all BOSH-deployed components in your deployment to trust a custom root certificate. If you want to use Docker Registries for running app instances in Docker containers, use this field to enter the certificate for your private Docker Registry. See the Using Docker Registries topic for more information.

3. Choose **Generate passwords** or **Use default BOSH password**. Pivotal recommends that you use the **Generate passwords** option for greater security.

4. Click **Save**. To view your saved Director password, click the **Credentials** tab.

Step 7: Syslog Page

1. Select **Syslog**.



2. (Optional) To send BOSH Director system logs to a remote server, select **Yes**.

3. In the **Address** field, enter the IP address or DNS name for the remote server.

4. In the **Port** field, enter the port number that the remote server listens on.

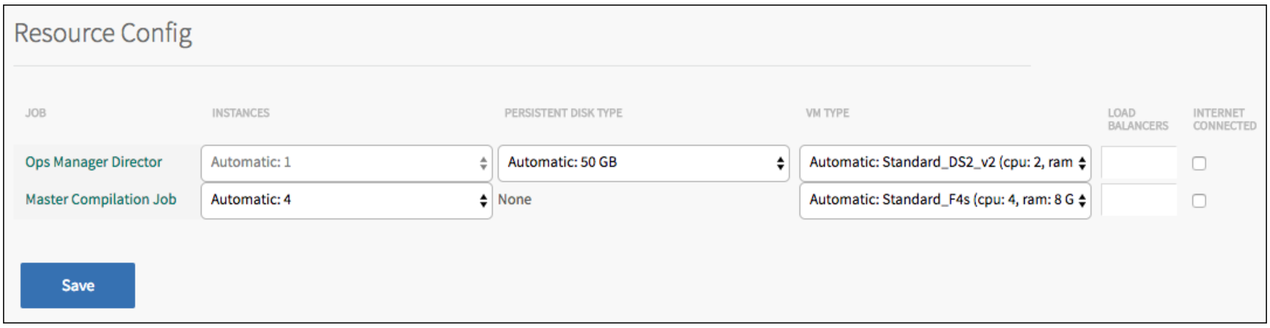
5. In the **Transport Protocol** dropdown menu, select **TCP**, **UDP**, or **RELP**. This selection determines which transport protocol is used to send the logs to the remote server.

6. (Optional) Mark the **Enable TLS** checkbox to use TLS encryption when sending logs to the remote server.
   - In the **Permitted Peer** field, enter either the name or SHA1 fingerprint of the remote peer.
   - In the **SSL Certificate** field, enter the SSL certificate for the remote server.

7. Click **Save**.

Step 8: Resource Config Page

1. Select **Resource Config**.



2. Ensure that the **Internet Connected** checkboxes are deselected for all jobs.

3. Adjust any values as necessary for your deployment. Under the **Instances**, **Persistent Disk Type**, and **VM Type** fields, choose **Automatic** from the drop-down menu to allocate the recommended resources for the job. If the **Persistent Disk Type** field reads **None**, the job does not require persistent disk space.

   Note: Ops Manager requires a Director VM with at least 8 GB memory.

   Note: If you set a field to **Automatic** and the recommended resource allocation changes in a future version, Ops Manager automatically uses the updated recommended allocation.

4. Click **Save**.

Step 9: Complete the Ops Manager Director Installation

1. Click **Apply Changes**. If a red ICMP error message appears and you have disabled ICMP, click **Ignore errors and start the install**.

2. Ops Manager Director installs. This may take a few moments. When the installation process successfully completes, the **Changes Applied** window appears.

3. Click the **Installation Dashboard** link to return to the Installation Dashboard.

4. When Ops Manager finishes deploying, you can optionally deploy BOSH Add-Ons to your system, and then install one or more PCF runtime environments to complete your PCF installation.

What to Do Next

After you complete this procedure, follow the instructions in the Deploying PAS on Azure topic.

Deploying PAS on Azure

Page last updated:

This topic describes how to install and configure Pivotal Application Service (PAS) on Azure.

Before you perform the procedures in this topic, you must have completed the procedures in the Preparing to Deploy PCF on Azure topic, either the Launching an Ops Manager Director Instance with an ARM Template topic or the Launching an Ops Manager Director Instance on Azure without an ARM Template topic, and the Configuring Ops Manager Director on Azure topic.

Note: If you plan to install the PCF IPsec add-on, you must do so before installing any other tiles. Pivotal recommends installing IPsec immediately after Ops Manager, and before installing the Pivotal Application Service (PAS) tile.

Note: The Azure portal sometimes displays the names of resources with incorrect capitalization. Always use the Azure CLI to retrieve the correctly capitalized name of a resource.

Step 1: Add PAS to Ops Manager

1. If you have not already downloaded PAS, log in to Pivotal Network, and click on PAS.

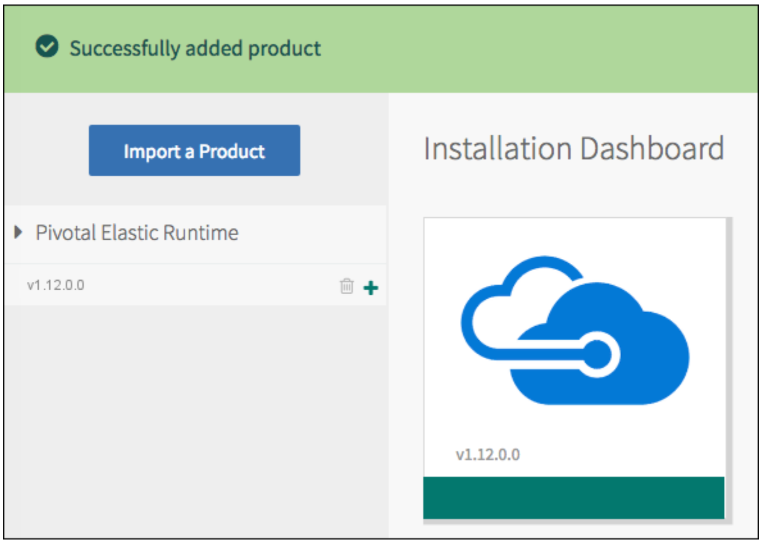
2. From the **Releases** drop-down, select the release to install and choose one of the following:

    a. Click PAS to download the PAS `.pivotal` file
    b. Click **PCF Small Footprint PAS** to download the Small Footprint Runtime `.pivotal` file. For more information, see Getting Started with Small Footprint Runtime.

3. Navigate to the Ops Manager Installation Dashboard.

4. Click **Import a Product** and select the downloaded `.pivotal` file. For more information, refer to the Adding and Deleting Products topic.

5. Click the plus button next to the imported tile to add it to the Installation Dashboard.

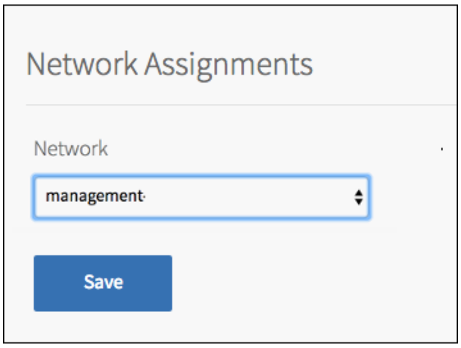


6. Click the PAS tile in the Installation Dashboard.

Pivotal

Step 2: Assign Networks

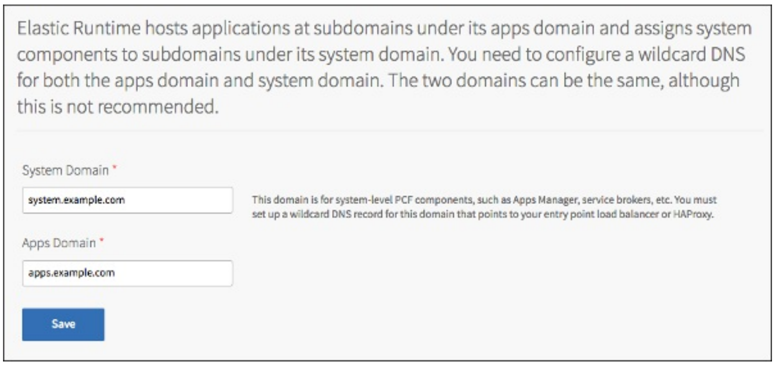
1. Select **Assign Networks**.
2. From the **Network** dropdown menu, select the network on which you want to run PAS.



3. Click **Save**.

Step 3: Configure Domains

1. Select **Domains**.



2. Enter the system and application domains.
   - The **System Domain** defines your target when you push apps to PAS. For example, `system.example.com`.
   - The **Apps Domain** defines where PAS should serve your apps. For example, `apps.example.com`.

> **Note:** Pivotal recommends that you use the same domain name but different subdomain names for your system and app domains. Doing so allows you to use a single wildcard certificate for the domain while preventing apps from creating routes that overlap with system routes.

3. Navigate to your DNS provider to create A records that point from your wildcard system and apps domains to the public IP address of your load balancer. For example, if the IP address of your load balancer is 198.51.100.1, then create an A record that points `*.system.example.com` to that address and another A record that points `*.apps.example.com` to that address.

> **Note:** To retrieve the IP address of your load balancer, navigate to the Azure portal, click **All resources**, and click the **Public IP address**

resource that ends with `pcf-lb-ip`.

4. Click **Save**.

Step 4: Configure Networking

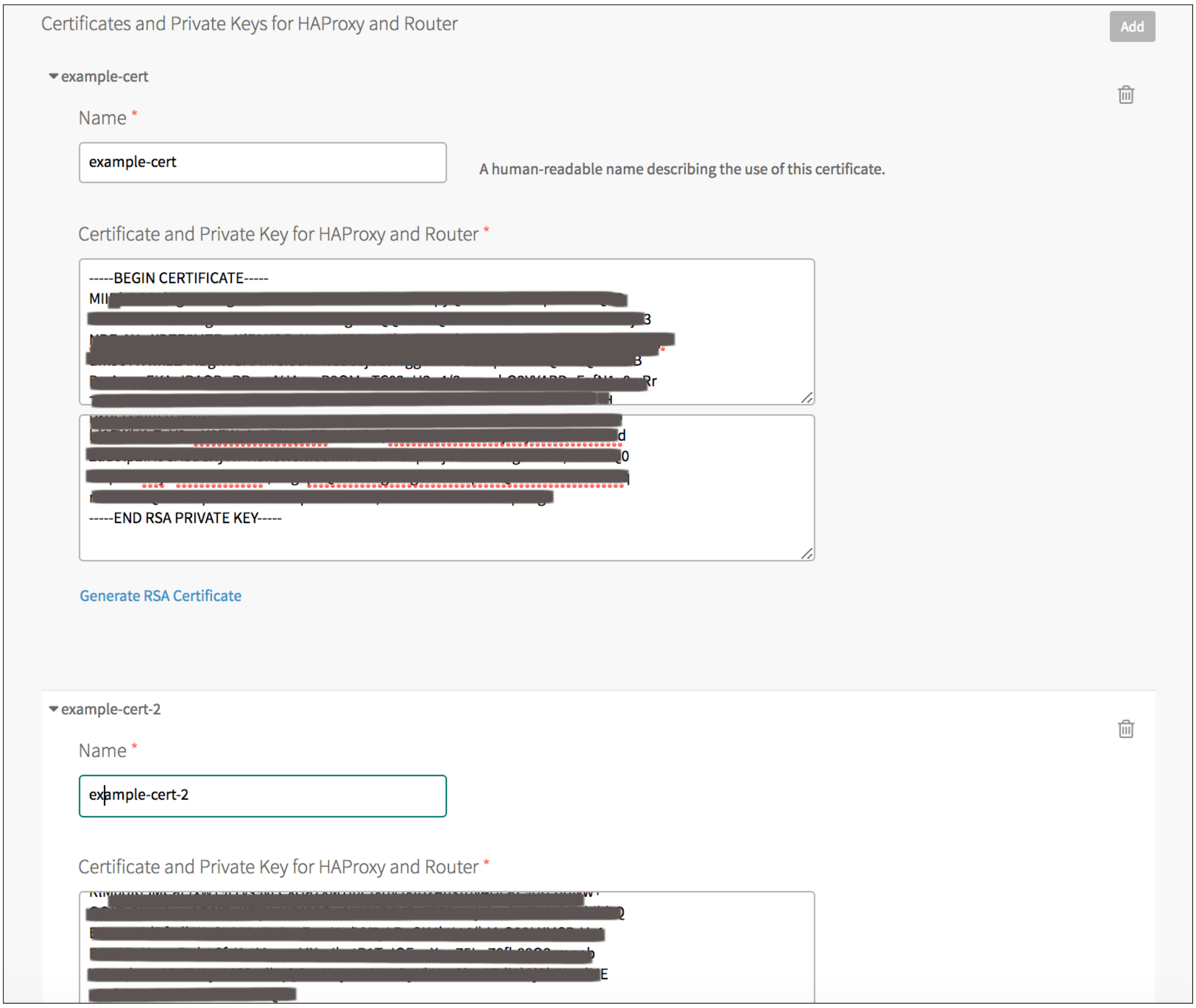
1. Select **Networking**.

2. Leave the **Router IPs**, **SSH Proxy IPs**, **HAProxy IPs**, and **TCP Router IPs** fields blank. You do not need to complete these fields when deploying PCF to Azure.

   > **Note**: You specify load balancers in the **Resource Config** section of PAS later on in the installation process. See the Configuring Resources section.

3. Under **Certificates and Private Key for HAProxy and Router**, you must provide at least one **Certificate and Private Key** name and certificate keypair for HAProxy and Gorouter. The HAProxy and Gorouter are enabled to receive TLS communication by default. You can configure multiple certificates for HAProxy and Gorouter.

   a. Click the **Add** button to add a name for the certificate chain and its private keypair. This certificate is the default used by Gorouter and HAProxy.

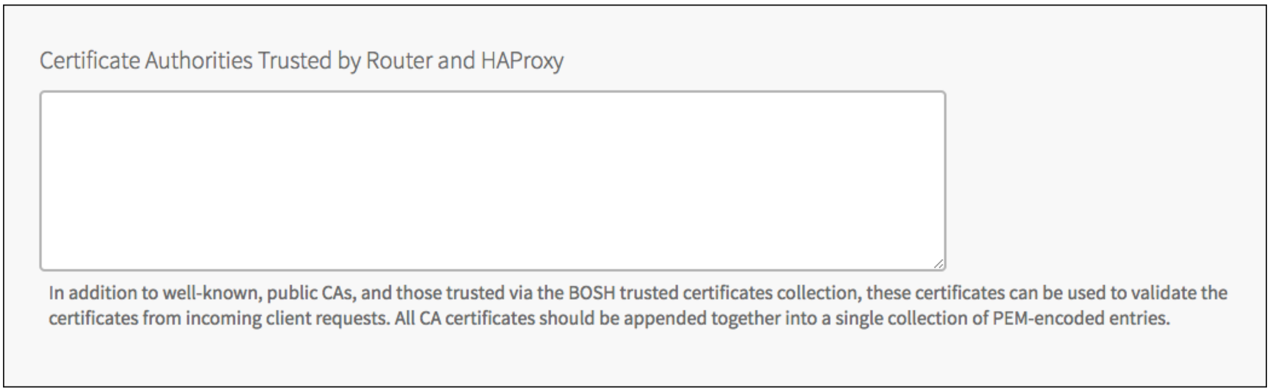


   You can either provide a certificate signed by a Certificate Authority (CA) or click on the **Generate RSA Certificate** link to generate a self-signed certificate in Ops Manager.

   b. If you want to configure multiple certificates for HAProxy and Gorouter, click the **Add** button and fill in the appropriate fields for each additional certificate keypair.

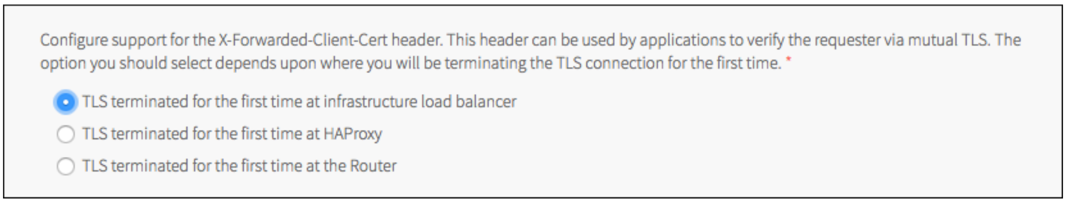
   For details about generating certificates in Ops Manager for your wildcard system domains, see the Providing a Certificate for Your SSL/TLS Termination Point topic.

4. (Optional) When validating client requests using mutual TLS, the Gorouter trusts multiple certificate authorities (CAs) by default. If you want to configure the Gorouter and HAProxy to trust additional CAs, enter your CA certificates under **Certificate Authorities Trusted by Router and HAProxy**. All CA certificates should be appended together into a single collection of PEM-encoded entries.



5. In the **Minimum version of TLS supported by HAProxy and Router** field, select the minimum version of TLS to use in HAProxy and Router communications. HAProxy and Router use TLS v1.2 by default. If you need to accommodate clients that use an older version of TLS, select a lower minimum version. For a list of TLS ciphers supported by the Gorouter, see [Securing Traffic into Cloud Foundry](#).

6. Under **Configure support for the X-Forwarded-Client-Cert header**, configure PCF handles `x-forwarded-client-cert` (XFCC) HTTP headers based on where TLS is terminated for the first time in your deployment.



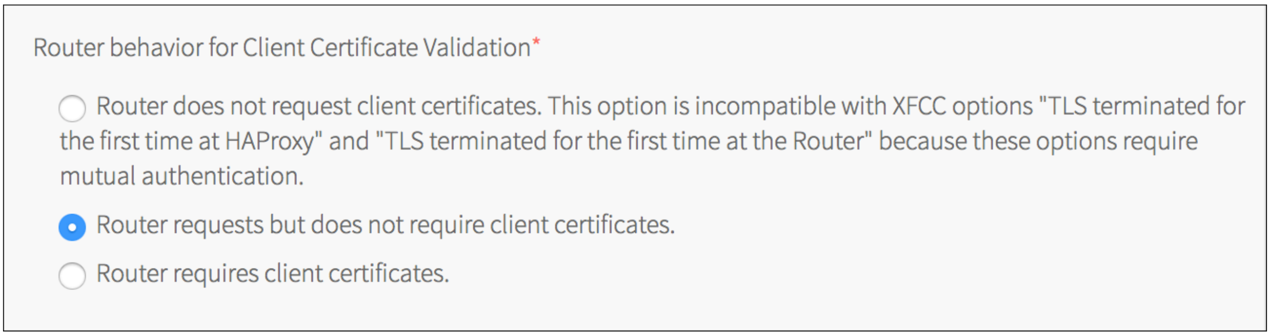
The following table indicates which option to choose based on your deployment layout.

If your deployment is configured as follows:	Then select the following option:	Additional notes:
◦ The Load Balancer is terminating TLS, and ◦ Load balancer is configured to put the client certificate from a mutual authentication TLS handshake into the X-Forwarded-Client-Cert HTTP header	TLS terminated for the first time at infrastructure load balancer (default).	Both HAProxy and the Gorouter forward the XFCC header when included in the request.
◦ The Load Balancer is configured to pass through the TLS handshake via TCP to the instances of HAProxy, and ◦ HAProxy instance count is > 0	TLS terminated for the first time at HAProxy.	HAProxy sets the XFCC header with the client certificate received in the TLS handshake. The Gorouter forwards the header. **Breaking Change**: In the **Router behavior for Client Certificates** field, you cannot select the **Router does not request client certificates** option.
◦ The Load Balancer is configured to pass		The Gorouter strips the XFCC header if it is included in the request and forwards the client certificate received in the TLS handshake in a new XFCC header. If you have deployed instances of HAProxy, app traffic bypasses those instances in this configuration. If you have also configured your load

through the TLS handshake via TCP to instances of the Gorouter	TLS terminated for the first time at the Gorouter.	balancer to route requests for ssh directly to the Diego Brain, consider reducing HAProxy instances to 0. **Breaking Change**: In the **Router behavior for Client Certificates** field, you cannot select the **Router does not request client certificates** option.

For a description of the behavior of each configuration option, see Forward Client Certificate to Applications.

7. To configure Gorouter behavior for handling client certificates, select one of the following options in the **Router behavior for Client Certificate Validation** field.



   - **Router does not request client certificates.** This option is incompatible with the XFCC configuration options **TLS terminated for the first time at HAProxy** and **TLS terminated for the first time at the Router** in PAS because these options require mutual authentication. As client certificates are not requested, client will not provide them, and thus validation of client certificates will not occur.
   - **Router requests but does not require client certificates.** Gorouter requests client certificates in TLS handshakes, validates them when presented, but does not require them. This is the default configuration.
   - **Router requires client certificates.** Gorouter validates that the client certificate is signed by a Certificate Authority that Gorouter trusts. If Gorouter cannot validate the client certificate, the TLS handshake fails.

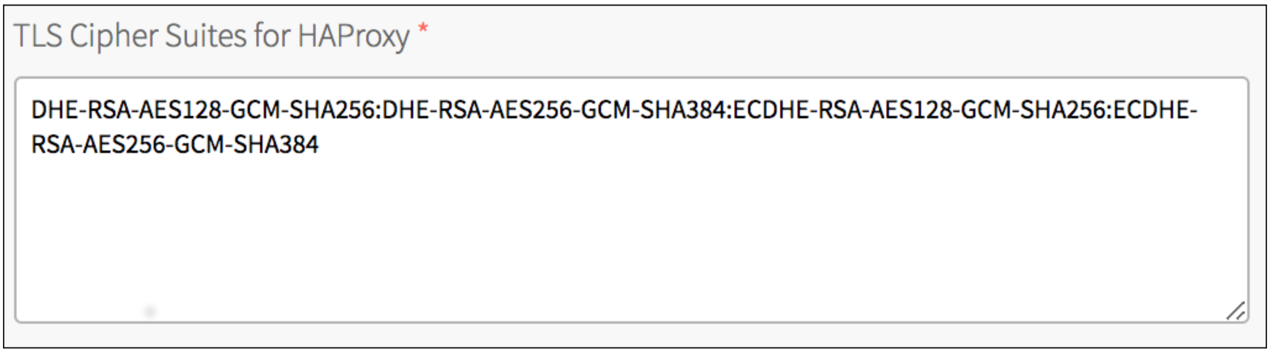
8. In the **TLS Cipher Suites for Router** field, specify the TLS cipher suites to use for TLS handshakes between the Gorouter and downstream clients like load balancers or HAProxy. Use an ordered, colon-delimited list of Golang-supported TLS cipher suites in the OpenSSL format. The recommended setting is `ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384`. Operators should verify that the ciphers are supported by any clients or downstream components that will initiate TLS handshakes with the Gorouter. For a list of TLS ciphers supported by the Gorouter, see Securing Traffic into Cloud Foundry.

   Verify that whatever client is participating in the TLS handshake with the Gorouter has at least one cipher suite in common with the Gorouter.

   > **Note:** Specify cipher suites that are supported by the versions configured in the **Minimum version of TLS supported by HAProxy and Router** field.

9. In the **TLS Cipher Suites for HAProxy** field, specify the TLS cipher suites to use in HAProxy for TLS handshakes between HAProxy and its clients such as load balancers and Gorouter. Use an ordered, colon-delimited list of TLS cipher suites in the OpenSSL format. The recommended setting: `DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384`

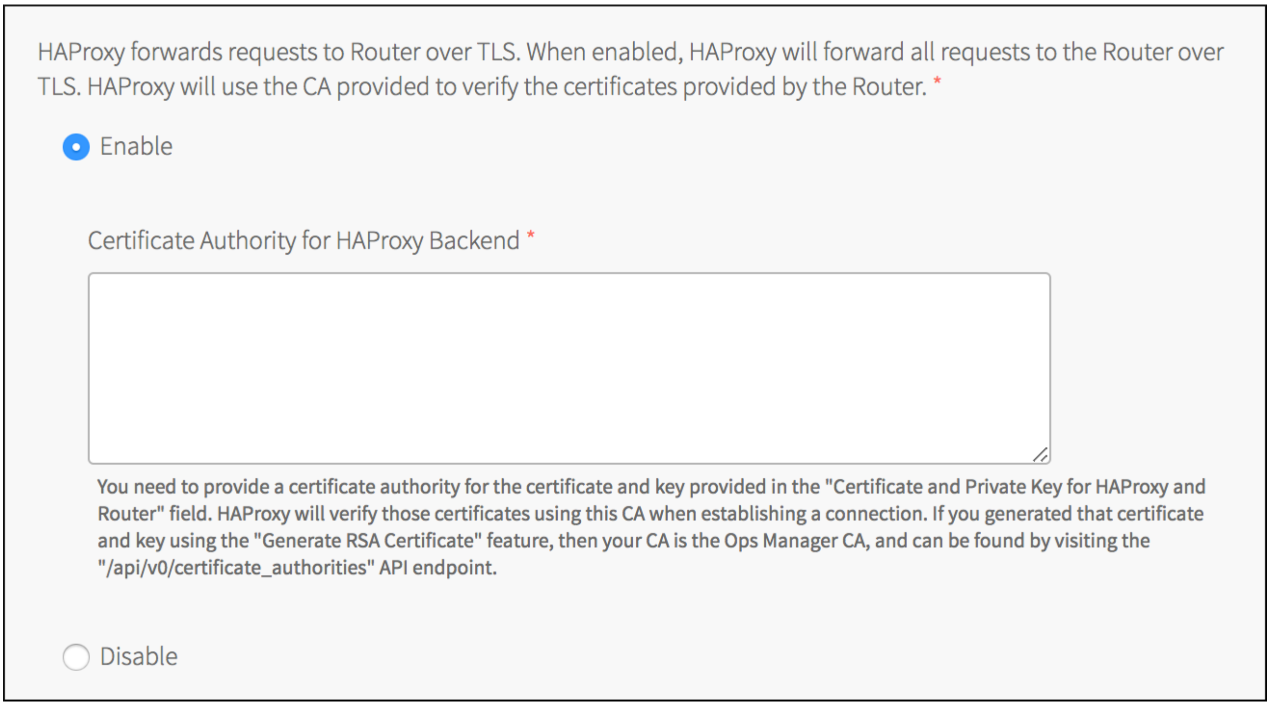
   Operators should verify that the ciphers are supported by any clients or downstream components that will initiate TLS handshakes with the HAProxy.



Verify that whatever clients are participating in the TLS handshake with HAProxy have at least one cipher suite in common with HAProxy.

> **Note:** Specify cipher suites that are supported by the versions configured in the **Minimum version of TLS supported by HAProxy and Router** field.

10. Under **HAProxy forwards requests to Router over TLS**, select **Enable** or **Disable** based on your deployment layout.



- **Enable HAProxy forwarding of requests to Router over TLS**

If you want to:	Encrypt communication between HAProxy and the Gorouter
Then configure the following:	1. Leave **Enable** selected. 2. In the **Certificate Authority for HAProxy Backend** field, specify the Certificate Authority (CA) that signed the certificate you configured in the **Certificate and Private Key for HAProxy and Router** field. **Note:** If you used the **Generate RSA Certificate** link to generate a self-signed certificate, then the CA to specify is the Ops Manager CA, which you can locate at the CA endpoint in the Ops Manager API. 3. Make sure that Gorouter and HAProxy have TLS cipher suites in common in the **TLS Cipher Suites for Router** and **TLS Cipher Suites for HAProxy** fields.
See also:	- Terminating SSL/TLS at the Load Balancer and Gorouter - Providing a Certificate for Your SSL/TLS Termination Point - Using the Ops Manager API

- **Disable HAProxy forwarding of requests to Router over TLS**

If you want to:	Use non-encrypted communication between HAProxy and Gorouter, or you are not using HAProxy

Then configure the following:	1. Select **Disable**. 2. If you are not using HAProxy, set the number of HAProxy job instances to `0` on the **Resource Config** page. See [Disable Unused Resources]().
See also:	◦ [Terminating SSL/TLS at the Gorouter Only]() ◦ [Terminating SSL/TLS at the Load Balancer Only]()

11. If you are not using SSL encryption or if you are using self-signed certificates, select **Disable SSL certificate verification for this environment**. Selecting this checkbox also disables SSL verification for route services.

 Note: For production deployments, Pivotal does not recommend disabling SSL certificate verification.

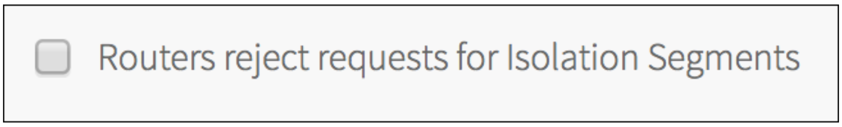
12. (Optional) If you want HAProxy or the Gorouter to reject any HTTP (non-encrypted) traffic, select the **Disable HTTP on HAProxy and Gorouter** checkbox. When selected, HAProxy and Gorouter will not listen on port 80.

13. (Optional) Select the **Disable insecure cookies on the Router** checkbox to set the secure flag for cookies generated by the router.

14. (Optional) To disable the addition of Zipkin tracing headers on the Gorouter, deselect the **Enable Zipkin tracing headers on the router** checkbox. Zipkin tracing headers are enabled by default. For more information about using Zipkin trace logging headers, see [Zipkin Tracing in HTTP Headers]().

15. (Optional) To stop the Router from writing access logs to local disk, deselect the **Enable Router to write access logs locally** checkbox. You should consider disabling this checkbox for high traffic deployments since logs may not be rotated fast enough and can fill up the disk.

16. By default, the PAS routers handle traffic for applications deployed to an isolation segment created by the PCF Isolation Segment tile. To configure the PAS routers to reject requests for applications within isolation segments, select the **Routers reject requests for Isolation Segments** checkbox.



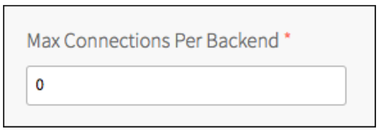
Do not enable this option without deploying routers for each isolation segment. See the following topics for more information:
    - [Installing PCF Isolation Segment]()
    - [Sharding Routers for Isolation Segments]()

17. In the **Choose whether or not to enable route services** section, choose either **Enable route services** or **Disable route services**. Route services are a class of [marketplace services]() that perform filtering or content transformation on application requests and responses. See the [Route Services]() topic for details.

18. (Optional) If you want to limit the number of app connections to the backend, enter a value in the **Max Connections Per Backend** field. You can use this field to prevent a poorly behaving app from all the connections and impacting other apps.

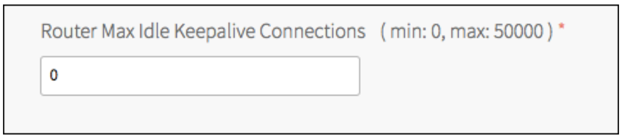
    To choose a value for this field, review the peak concurrent connections received by instances of the most popular apps in your deployment. You can determine the number of concurrent connections for an app from the `httpStartStop` event metrics emitted for each app request.

    If your deployment uses PCF Metrics, you can also obtain this peak concurrent connection information from [Network Metrics](). The default value is



`500`.

19. Enter a value for **Router Max Idle Keepalive Connections**. See [Considerations for Configuring max_idle_connections]().



20. (Optional) Increase the number of seconds in the **Router Timeout to Backends** field to accommodate larger uploads over connections with high latency. Set this value to less than or equal to the idle timeout value of the Azure load balancer, which defaults to 4 minutes.

> **Note:** If the router timeout value exceeds the Azure LB timeout, you may experience intermittent TCP resets. For more information about configuring Azure load balancer idle timeout, see the Azure documentation.

21. (Optional) Use the **Frontend Idle Timeout for Gorouter and HAProxy** field to help prevent connections from your load balancer to Gorouter or HAProxy from being closed prematurely. The value you enter sets the duration, in seconds, that Gorouter or HAProxy maintains an idle open connection from a load balancer that supports keep-alive.

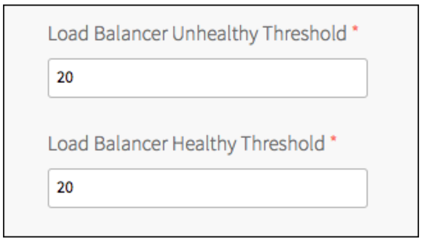
    In general, set the value higher than your load balancer's backend idle timeout to avoid the race condition where the load balancer sends a request before it discovers that Gorouter or HAProxy has closed the connection.

    See the following table for specific guidance and exceptions to this rule:

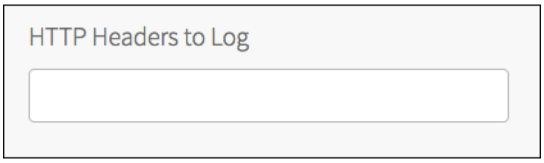
IaaS	Guidance
AWS	AWS ELB has a default timeout of 60 seconds, so Pivotal recommends a value greater than `60`.
Azure	By default, Azure load balancer times out at 240 seconds without sending a TCP RST to clients, so as an exception, Pivotal recommends a value lower than `240` to force the load balancer to send the TCP RST.
GCP	GCP has a default timeout of 600 seconds, so Pivotal recommends a value greater than `600`.
Other	Set the timeout value to be greater than that of the load balancer's backend idle timeout.

22. (Optional) Increase the value of **Load Balancer Unhealthy Threshold** to specify the amount of time, in seconds, that the router continues to accept connections before shutting down. During this period, healthchecks may report the router as unhealthy, which causes load balancers to failover to other routers. Set this value to an amount greater than or equal to the maximum time it takes your load balancer to consider a router instance unhealthy, given contiguous failed healthchecks.

23. (Optional) Modify the value of **Load Balancer Healthy Threshold**. This field specifies the amount of time, in seconds, to wait until declaring the Router instance started. This allows an external load balancer time to register the Router instance as healthy.

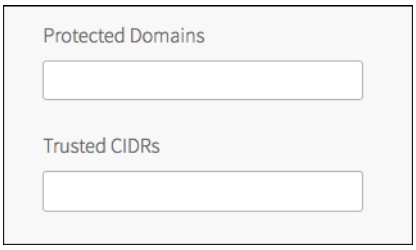


24. (Optional) If app developers in your organization want certain HTTP headers to appear in their app logs with information from the Gorouter, specify them in the **HTTP Headers to Log** field. For example, to support app developers that deploy Spring apps to PCF, you can enter Spring-specific HTTP headers.

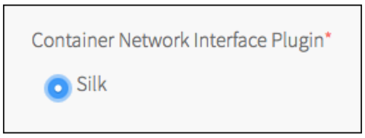


25. If you expect requests larger than the default maximum of 16 Kbytes, enter a new value (in bytes) for **HAProxy Request Max Buffer Size**. You may need to do this, for example, to support apps that embed a large cookie or query string values in headers.

26. If your PCF deployment uses HAProxy and you want it to receive traffic only from specific sources, use the following fields:
    - **Protected Domains:** Enter a comma-separated list of domains from which PCF can receive traffic.
    - **Trusted CIDRs:** Optionally, enter a space-separated list of CIDRs to limit which IP addresses from the **Protected Domains** can send traffic to PCF.



27. The **Loggregator Port** defaults to `443` if left blank. Leave this field blank.



28. For **Container Network Plugin Interface**, ensure **Silk** is selected and review the following fields:

> **Note:** The **External** option exists to support NSX-T integration for vSphere deployments.

   a. (Optional) You can change the value in the **Applications Network Maximum Transmission Unit (MTU)** field. Pivotal recommends setting the MTU value for your application network to `1454`. Some configurations, such as networks that use GRE tunnels, may require a smaller MTU value.
   b. (Optional) Enter an IP range for the overlay network in the **Overlay Subnet** box. If you do not set a custom range, Ops Manager uses `10.255.0.0/16`.

   > **WARNING**: The overlay network IP range must not conflict with any other IP addresses in your network.

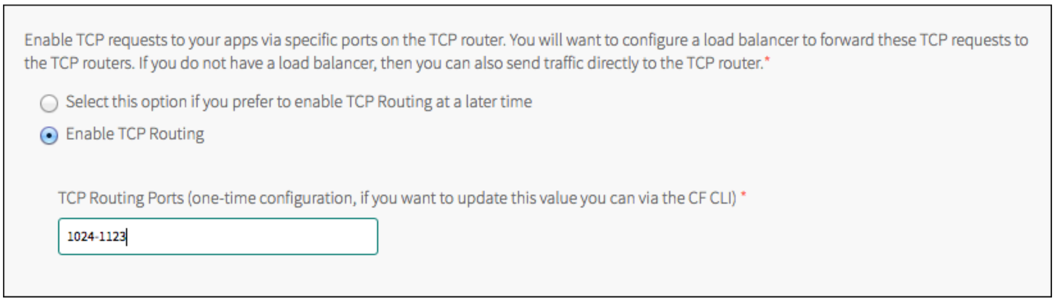
   c. Enter a UDP port number in the **VXLAN Tunnel Endpoint Port** box. If you do not set a custom port, Ops Manager uses 4789.
   d. For **Denied logging interval**, set the per-second rate limit for packets blocked by either a container-specific networking policy or by Application Security Group rules applied across the space, org, or deployment. This field defaults to `1`.
   e. For **UDP logging interval**, set the per-second rate limit for UDP packets sent and received. This field defaults to `100`.
   f. To enable logging for app traffic, select **Log traffic for all accepted/denied application packets**. See Manage Logging for Container-to-Container Networking for more information.

29. (Optional) TCP Routing is disabled by default. To enable this feature, perform the following steps:

   a. Select **Enable TCP Routing**.
   b. In **TCP Routing Ports**, enter a range of ports to be allocated for TCP Routes.

   For each TCP route you want to support, you must reserve a range of ports. This is the same range of ports you configured your load balancer with in the Pre-Deployment Steps, unless you configured DNS to resolve the TCP domain name to the TCP router directly.

   The **TCP Routing Ports** field accepts a comma-delimited list of individual ports and ranges, for example `1024-1099,30000,60000-60099`. Configuration of this field is only applied on the first deploy, and update updates to the port range are made using the cf CLI. For details about modifying the port range, see the Router Groups topic.



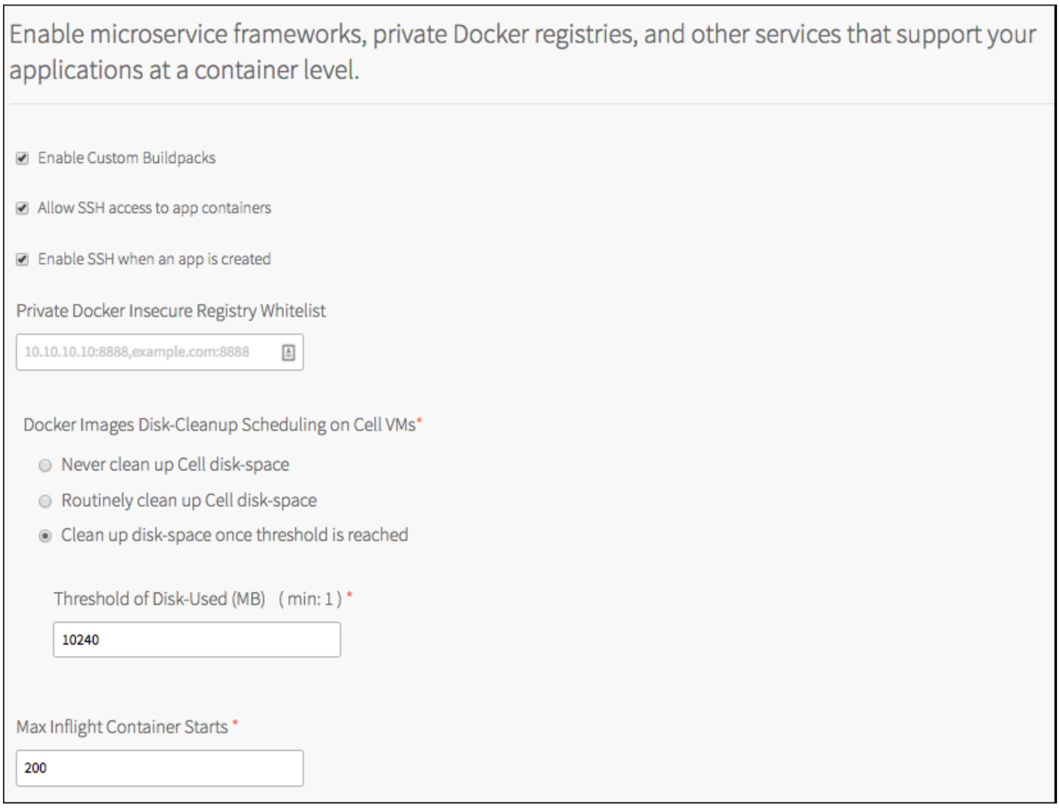
   c. For Azure, you also need to specify the name of Azure load balancer in the LOAD BALANCER column of TCP Router job of the **Resource Config** screen. You configure this later on in PAS. See Configuring Resources.

30. (Optional) To disable TCP routing, click **Select this option if you prefer to enable TCP Routing at a later time** For more information, see the Configuring TCP Routing in PAS topic.

31. Click **Save**.

Step 5: Configure Application Containers

1. Select **Application Containers**.



2. The **Enable Custom Buildpacks** checkbox governs the ability to pass a custom buildpack URL to the `-b` option of the `cf push` command. By default, this ability is enabled, letting developers use custom buildpacks when deploying apps. Disable this option by disabling the checkbox. For more information about custom buildpacks, refer to the buildpacks section of the PCF documentation.

3. The **Allow SSH access to app containers** checkbox controls SSH access to application instances. Enable the checkbox to permit SSH access across your deployment, and disable it to prevent all SSH access. See the Application SSH Overview topic for information about SSH access permissions at the space and app scope.

4. If you want enable SSH access for new apps by default in spaces that allow SSH, select **Enable SSH when an app is created**. If you deselect the checkbox, developers can still enable SSH after pushing their apps by running `cf enable-ssh APP-NAME`.

5. You can configure Pivotal Application Service (PAS) to run app instances in Docker containers by supplying their IP address range(s) in the **Private Docker Insecure Registry Whitelist** textbox. See the Using Docker Registries topic for more information.

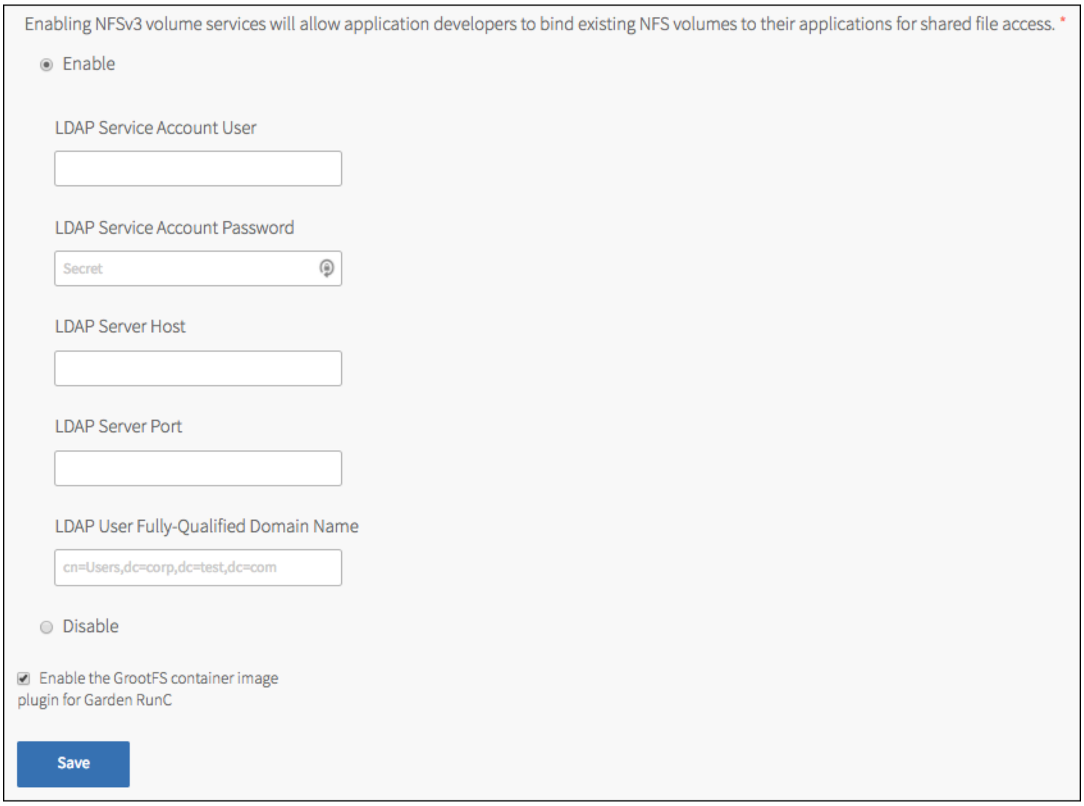
6. Select your preference for **Docker Images Disk-Cleanup Scheduling on Cell VMs**. If you choose **Clean up disk-space once threshold is reached**, enter a **Threshold of Disk-Used** in megabytes. For more information about the configuration options and how to configure a threshold, see Configuring Docker Images Disk-Cleanup Scheduling.

7. Enter a number in the **Max Inflight Container Starts** textbox. This number configures the maximum number of started instances across your deployment's Diego Cells. For more information about this feature, see Setting a Maximum Number of Started Containers.

8. Under **Enabling NFSv3 volume services**, select **Enable** or **Disable**. NFS volume services allow application developers to bind existing NFS volumes to their applications for shared file access. For more information, see the Enabling NFS Volume Services topic.

   > **Note**: In a clean install, NFSv3 volume services is enabled by default. In an upgrade, NFSv3 volume services is set to the same setting as it was in the previous deployment.

9. (Optional) To configure LDAP for NFSv3 volume services, perform the following steps:



- For **LDAP Service Account User**, enter the username of the service account in LDAP that will manage volume services.
- For **LDAP Service Account Password**, enter the password for the service account.
- For **LDAP Server Host**, enter the hostname or IP address of the LDAP server.
- For **LDAP Server Port**, enter the LDAP server port number. If you do not specify a port number, Ops Manager uses 389.
- For **LDAP Server Protocol**, enter the server protocol. If you do not specify a protocol, Ops Manager uses TCP.
- For **LDAP User Fully-Qualified Domain Name**, enter the fully qualified path to the LDAP service account. For example, if you have a service account named `volume-services` that belongs to organizational units (OU) named `service-accounts` and `my-company`, and your domain is named `domain`, the fully qualified path looks like the following:

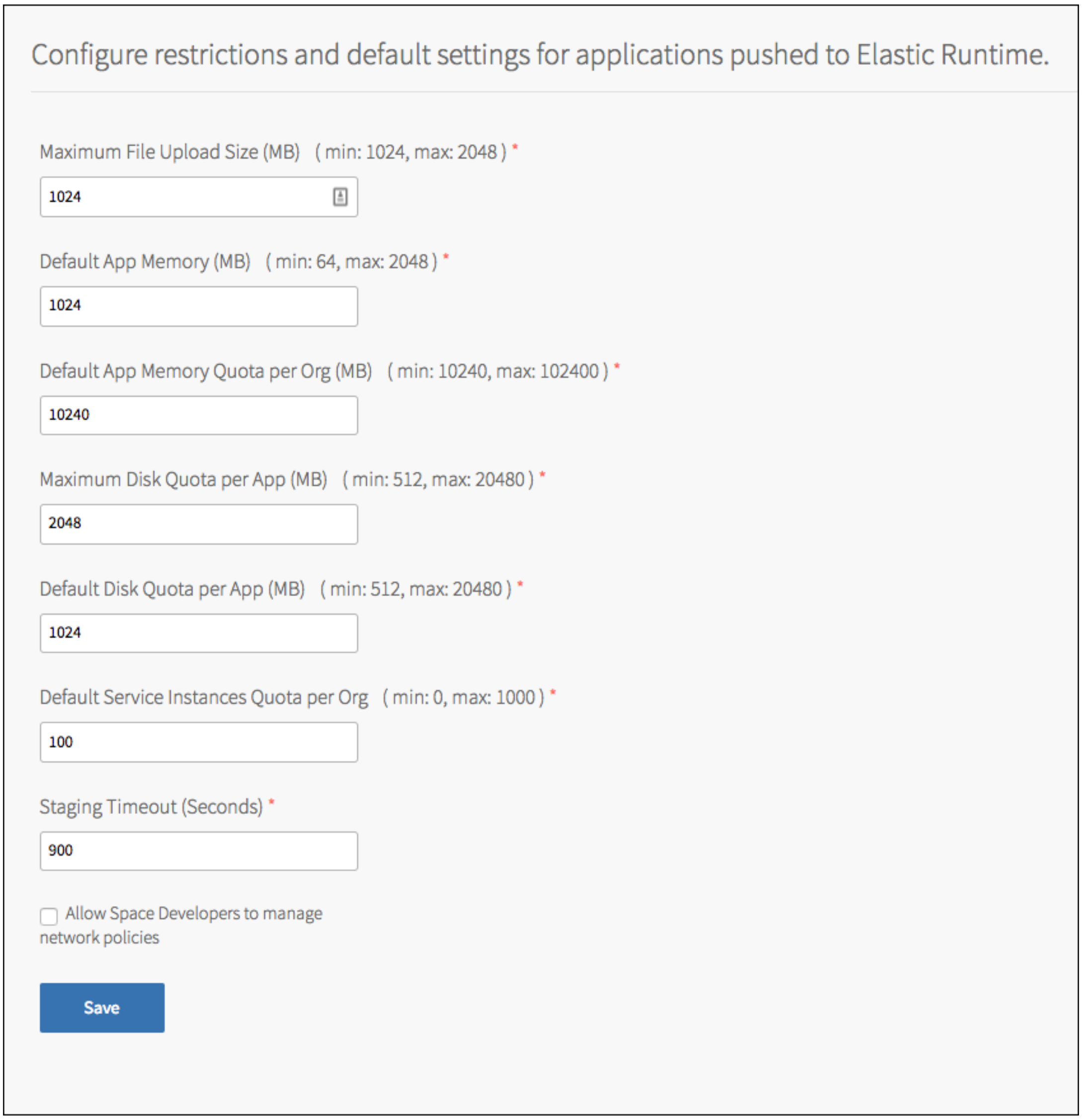
  ```
  CN=volume-services,OU=service-accounts,OU=my-company,DC=domain,DC=com
  ```

10. By default, PAS manages container images using the GrootFS plugin for Garden-runC. If you experience issues with GrootFS, you can disable the plugin and use the image plugin built into Garden-runC.

11. Click **Save**.

Step 6: Configure Application Developer Controls

1. Select **Application Developer Controls**.



2. Enter the **Maximum File Upload Size (MB)**. This is the maximum size of an application upload.

3. Enter the **Default App Memory (MB)**. This is the amount of RAM allocated by default to a newly pushed application if no value is specified with the cf CLI.

4. Enter the **Default App Memory Quota per Org** This is the default memory limit for all applications in an org. The specified limit only applies to the first installation of PAS. After the initial installation, operators can use the cf CLI to change the default value.

5. Enter the **Maximum Disk Quota per App (MB)**. This is the maximum amount of disk allowed per application.

> **Note**: If you allow developers to push large applications, PAS may have trouble placing them on Cells. Additionally, in the event of a system upgrade or an outage that causes a rolling deploy, larger applications may not successfully re-deploy if there is insufficient disk capacity. Monitor your deployment to ensure your Cells have sufficient disk to run your applications.

6. Enter the **Default Disk Quota per App (MB)**. This is the amount of disk allocated by default to a newly pushed application if no value is specified with the cf CLI.

7. Enter the **Default Service Instances Quota per Org** The specified limit only applies to the first installation of PAS. After the initial installation, operators can use the cf CLI to change the default value .

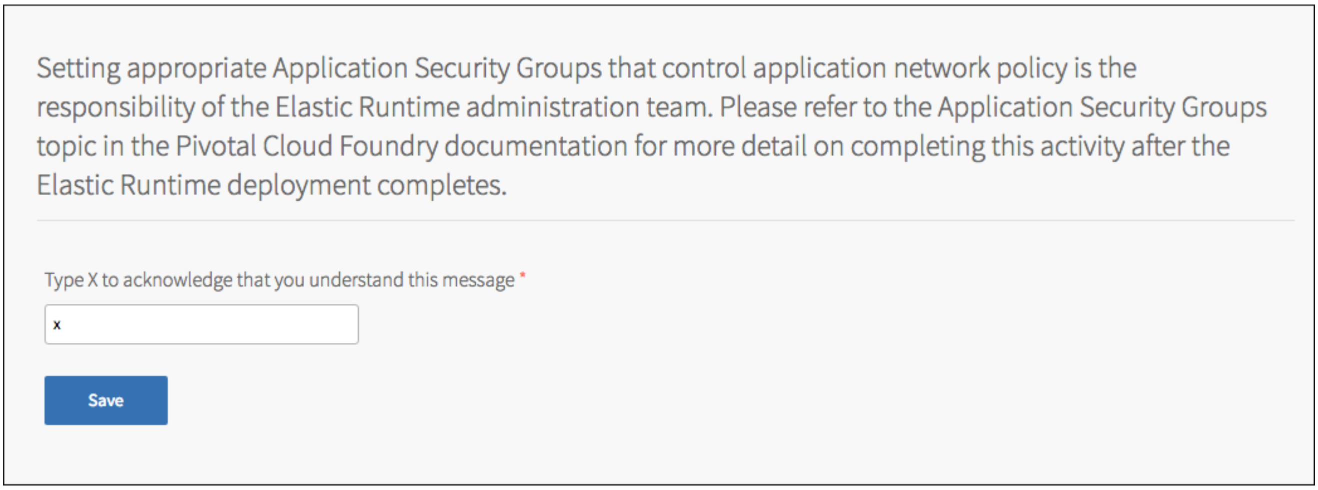
8. Enter the **Staging Timeout (Seconds)**. When you stage an application droplet with the Cloud Controller, the server times out after the number of seconds you specify in this field.

9. Select the **Allow Space Developers to manage network policies** checkbox to permit developers to manage their own network policies for their applications.

10. Click **Save**.

Step 7: Review Application Security Group

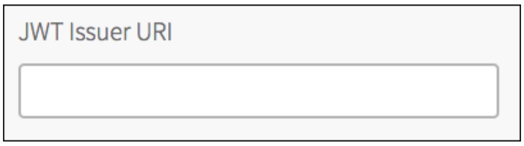
Setting appropriate Application Security Groups is critical for a secure deployment. Type `X` in the box to acknowledge that once the Pivotal Application Service (PAS) deployment completes, you will review and set the appropriate application security groups. See Restricting App Access to Internal PCF Components for instructions.



Step 8: Configure UAA

1. Select **UAA**.

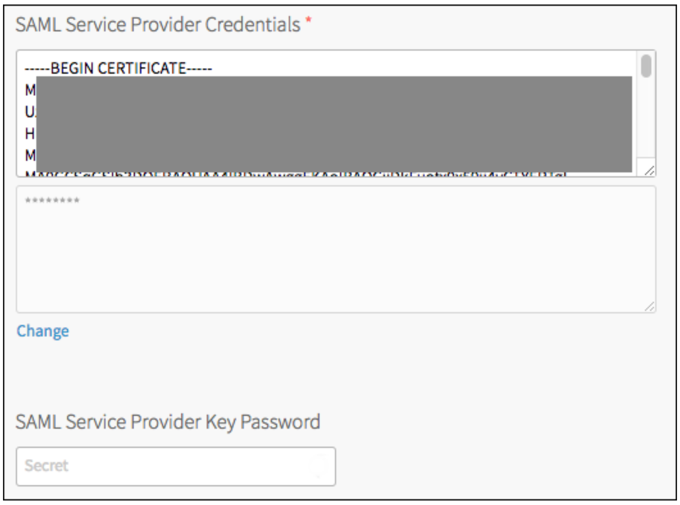
2. (Optional) Under **JWT Issuer URI**, enter the URI that UAA uses as the issuer when generating tokens.



3. Under **SAML Service Provider Credentials**, enter a certificate and private key to be used by UAA as a SAML Service Provider for signing outgoing SAML authentication requests. You can provide an existing certificate and private key from your trusted Certificate Authority or generate a self-signed certificate. The following domain must be associated with the certificate: `*.login.YOUR-SYSTEM-DOMAIN`.

> **Note**: The Pivotal Single Sign-On Service and Pivotal Spring Cloud Services tiles require the `*.login.YOUR-SYSTEM-DOMAIN`.

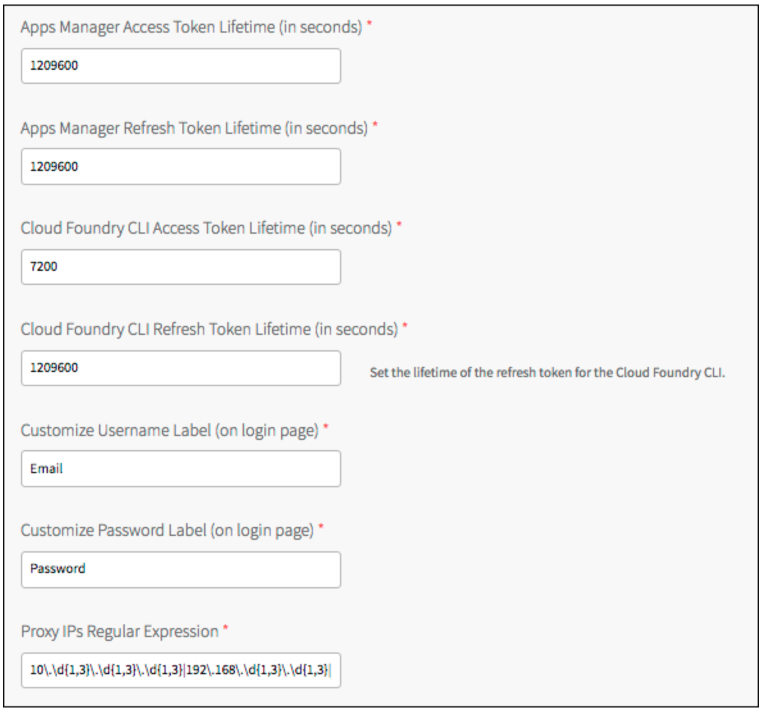
4. If the private key specified under **Service Provider Credentials** is password-protected, enter the password under **SAML Service Provider Key**



Password.

5. For **Signature Algorithm**, choose an algorithm from the dropdown menu to use for signed requests and assertions. The default value is `SHA256`.

6. (Optional) In the **Apps Manager Access Token Lifetime**, **Apps Manager Refresh Token Lifetime**, **Cloud Foundry CLI Access Token Lifetime**, and **Cloud Foundry CLI Refresh Token Lifetime** fields, change the lifetimes of tokens granted for Apps Manager and Cloud Foundry Command Line Interface (cf CLI) login access and refresh. Most deployments use the defaults.



7. (Optional) Customize the text prompts used for username and password from the cf CLI and Apps Manager login popup by entering values for **Customize Username Label (on login page)** and **Customize Password Label (on login page)**.

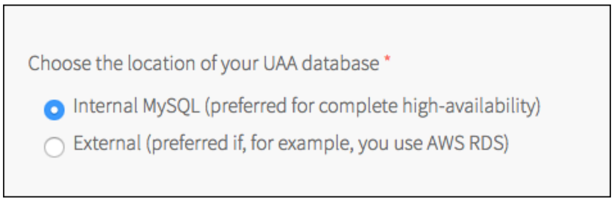
8. (Optional) The **Proxy IPs Regular Expression** field contains a pipe-delimited set of regular expressions that UAA considers to be reverse proxy IP addresses. UAA respects the `x-forwarded-for` and `x-forwarded-proto` headers coming from IP addresses that match these regular expressions. To configure UAA to respond properly to Gorouter or HAProxy requests coming from a public IP address, append a regular expression or regular expressions to match the public IP address.

9. You can configure UAA to use the internal MySQL database provided with PCF, or you can configure an external database provider. Follow the procedures in either the Internal Database Configuration or the External Database Configuration section below.

Note: If you are performing an upgrade, do not modify your existing internal database configuration or you may lose data. You must migrate your existing data before changing the configuration. See Upgrading Pivotal Cloud Foundry for additional upgrade information, and contact Pivotal Support for help.

Internal Database Configuration

1. Select **Internal MySQL**.

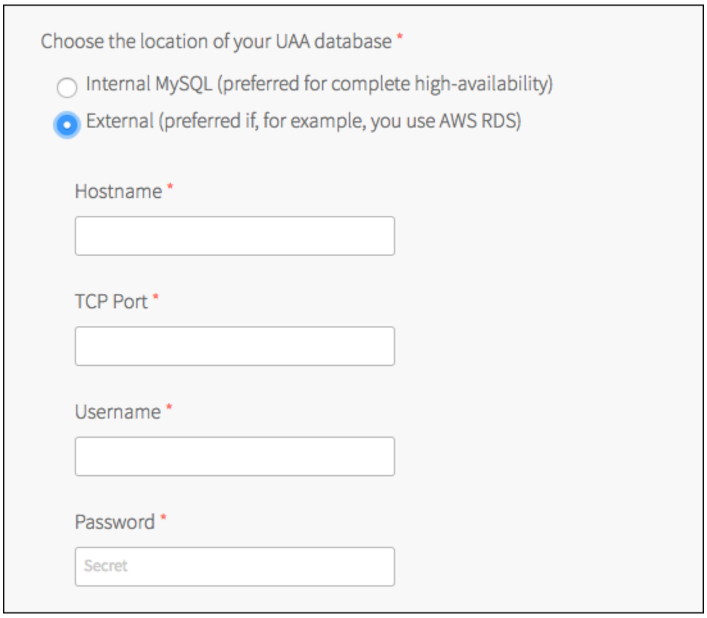


2. Click **Save**.

3. Ensure that you complete the “Configure Internal MySQL” step later in this topic to configure high availability and automatic backups for your internal MySQL databases.

External Database Configuration

1. From the **UAA** section in Pivotal Application Service (PAS), select **External**.



2. For **Hostname**, enter the hostname of the database server.

3. For **TCP Port**, enter the port of the database server.

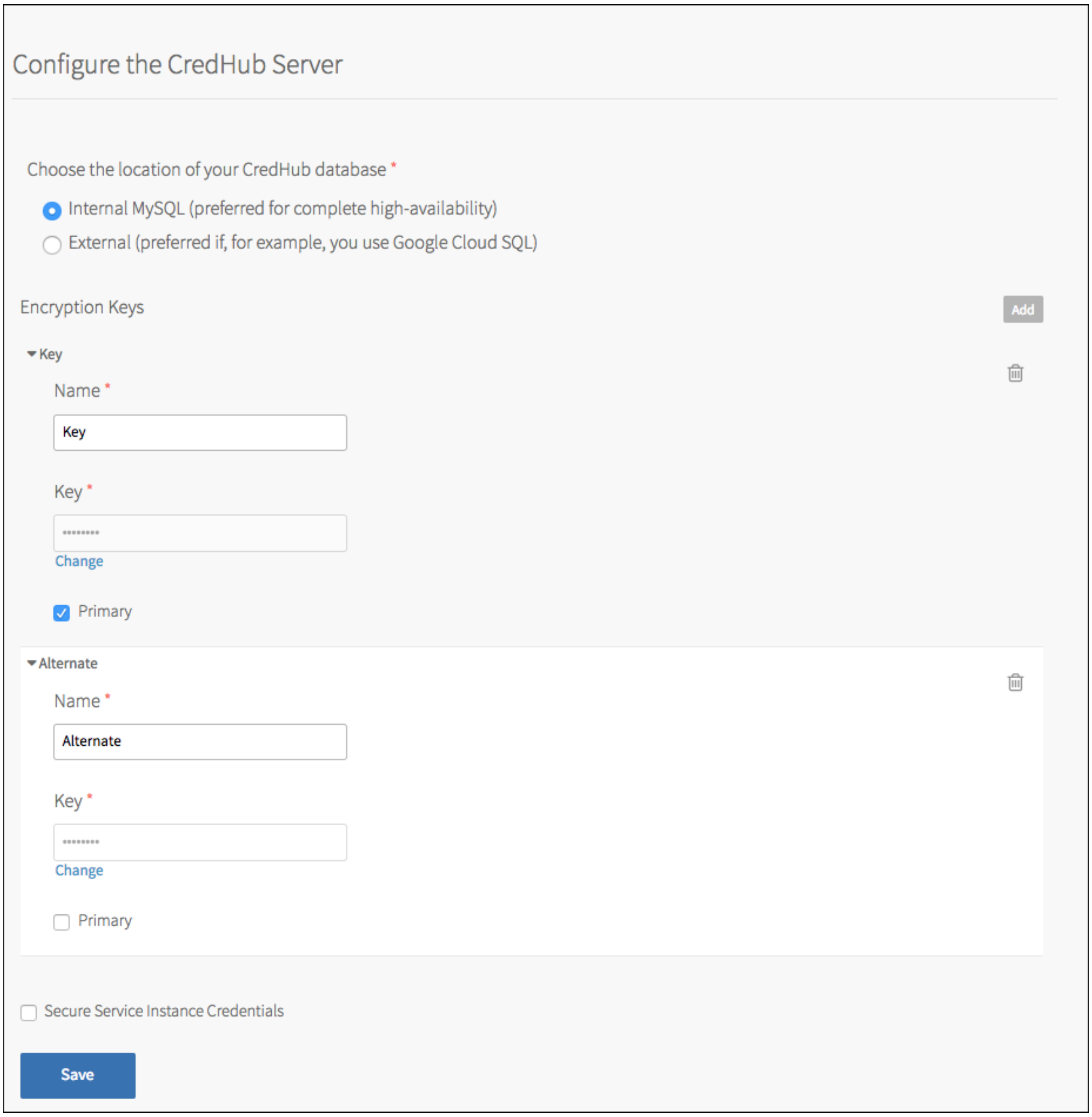
4. For **User Account and Authentication database username**, specify a unique username that can access this specific database on the database server.

5. For **User Account and Authentication database password**, specify a password for the provided username.

6. Click **Save**.

Step 9: (Optional) Configure CredHub

1. Select **Credhub**.



2. Choose the location of your CredHub Database. PAS includes this CredHub database for services to store their service instance credentials.

    a. If you chose **External**, enter the following:
        - **Hostname**: The IP address of your database server.
        - **TCP Port**: The port of your database server, such as `3306`.
        - **Username**: A unique username that can access this specific database on the database server.
        - **Password**: The password for the provided username.
        - **Database CA Certificate**: Enter a certificate to use for encrypting traffic to and from the database.

3. Under **Encryption Keys**, specify a key to use for encrypting and decrypting the values stored in the CredHub database.
    - **Name**: Enter the name of the key.
    - **Key**: Enter a key that is at least 20 characters in length.
    - **Primary**: Select this checkbox to use this key as your primary key.

    > **Note**: Ensure that you only mark one key as **Primary**. The UI includes an **Add** button to add more keys to support key rotation. For more information, see the Rotating PAS CredHub Encryption Keys topic.

4. If your deployment uses any PCF services that support storing service instance credentials in CredHub and you want to enable this feature, select the **Secure Service Instance Credentials** checkbox.

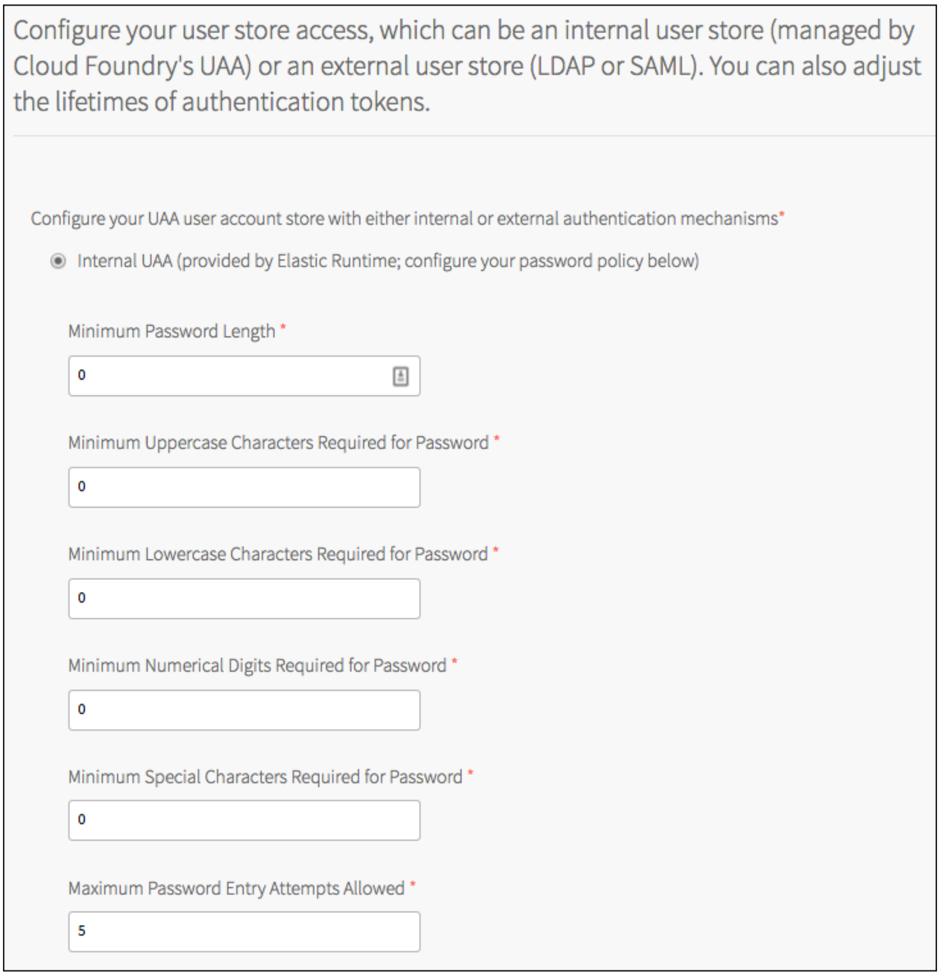
5. Select the **Resource Config** pane.

6. Under the **Job** column of the **CredHub** row, set the number of instances to `2`. This is the minimum instance count required for high availability.

Note: To use the Runtime CredHub feature, you must follow the additional steps in Securing Service Instance Credentials with Runtime CredHub.

Step 10: Configure Authentication and Enterprise SSO

1. Select **Authentication and Enterprise SSO**.



2. To authenticate user sign-ons, your deployment can use one of three types of user database: the UAA server's internal user store, an external SAML identity provider, or an external LDAP server.
   - To use the internal UAA, select the **Internal** option and follow the instructions in the Configuring UAA Password Policy topic to configure your password policy.
   - To connect to an external identity provider through SAML, scroll down to select the **SAML Identity Provider** option and follow the instructions in the Configuring PCF for SAML section of the *Configuring Authentication and Enterprise SSO for Pivotal Application Service (PAS)* topic.
   - To connect to an external LDAP server, scroll down to select the **LDAP Server** option and follow the instructions in the Configuring LDAP section of the *Configuring Authentication and Enterprise SSO for PAS* topic.

3. Click **Save**.

Step 11: Configure System Databases

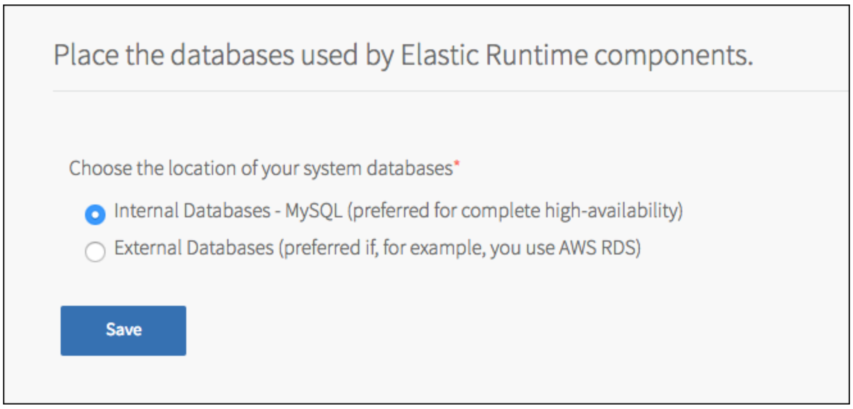
You can configure PAS to use the internal MySQL database provided with PCF, or you can configure an external database provider for the databases required by PAS.

Note: If you are performing an upgrade, do not modify your existing internal database configuration or you may lose data. You must migrate your existing data first before changing the configuration. Contact Pivotal Support for help. See Upgrading Pivotal Cloud Foundry for additional upgrade information.

Internal Database Configuration

If you want to use internal databases for your deployment, perform the following steps:

1. Select **Databases**.



2. Select **Internal Databases - MySQL**.
3. Click **Save**.

Then proceed to Step 12: (Optional) Configure Internal MySQL to configure high availability and automatic backups for your internal MySQL databases.

External Database Configuration

Note: To configure an external database for UAA, see the *External Database Configuration* section of Configure UAA.

Note: The exact procedure to create databases depends upon the database provider you select for your deployment. The following procedure uses AWS RDS as an example. You can configure a different database provider that provides MySQL support, such as Google Cloud SQL.

Warning: Protect whichever database you use in your deployment with a password.

To create your Pivotal Application Service (PAS) databases, perform the following steps:

1. Add the `ubuntu` account key pair from your IaaS deployment to your local SSH profile so you can access the Ops Manager VM. For example, in AWS, you add a key pair created in AWS:

```
$ ssh-add aws-keypair.pem
```

2. SSH in to your Ops Manager using the Ops Manager FQDN and the username `ubuntu`:

```
$ ssh ubuntu@OPS-MANAGER-FQDN
```

3. Log in to your MySQL database instance using the appropriate hostname and user login values configured in your IaaS account. For example, to log in to your AWS RDS instance, run the following MySQL command:

```
$ mysql --host=RDSHOSTNAME --user=RDSUSERNAME --password=RDSPASSWORD
```

4. Run the following MySQL commands to create databases for the eleven PAS components that require a relational database:

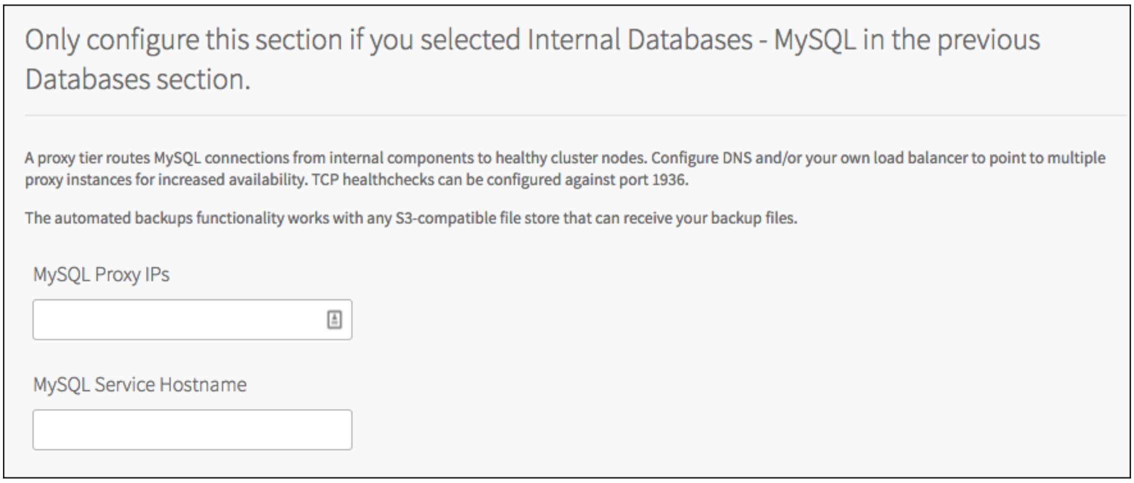
```
CREATE database ccdb;
CREATE database notifications;
CREATE database autoscale;
CREATE database app_usage_service;
CREATE database routing;
CREATE database diego;
CREATE database account;
CREATE database nfsvolume;
CREATE database networkpolicyserver;
CREATE database silk;
CREATE database locket;
```

5. Type `exit` to quit the MySQL client, and `exit` again to close your connection to the Ops Manager VM.
6. In PAS, select **Databases**.
7. Select the **External Databases** option.
8. For **Hostname**, enter the hostname of the database server.
9. For **TCP Port**, enter the port of the database server.
10. Each component that requires a relational database has two corresponding fields: one for the database username and one for the database password. For each set of fields, specify a unique username that can access this specific database on the database server and a password for the provided username.
11. Click **Save**.

Step 12: (Optional) Configure Internal MySQL

Note: You only need to configure this section if you have selected **Internal Databases - MySQL** in the **Databases** section.

1. Select **Internal MySQL**.
2. In the **MySQL Proxy IPs** field, enter one or more comma-delimited IP addresses that are not in the reserved CIDR range of your network. If a MySQL node fails, these proxies re-route connections to a healthy node. See the MySQL Proxy topic for more information.



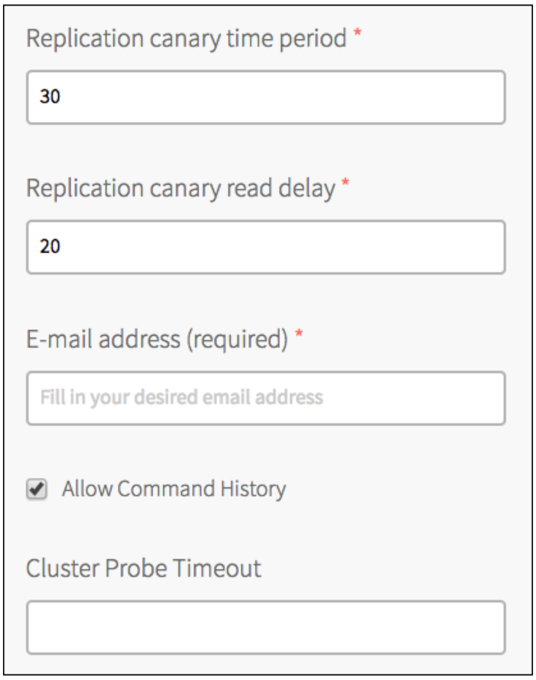
3. For **MySQL Service Hostname**, enter an IP address or hostname for your load balancer. If a MySQL proxy fails, the load balancer re-routes connections to a healthy proxy. If you leave this field blank, components are configured with the IP address of the first proxy instance entered above.

   Warning: You must configure a load balancer to achieve complete high-availability.

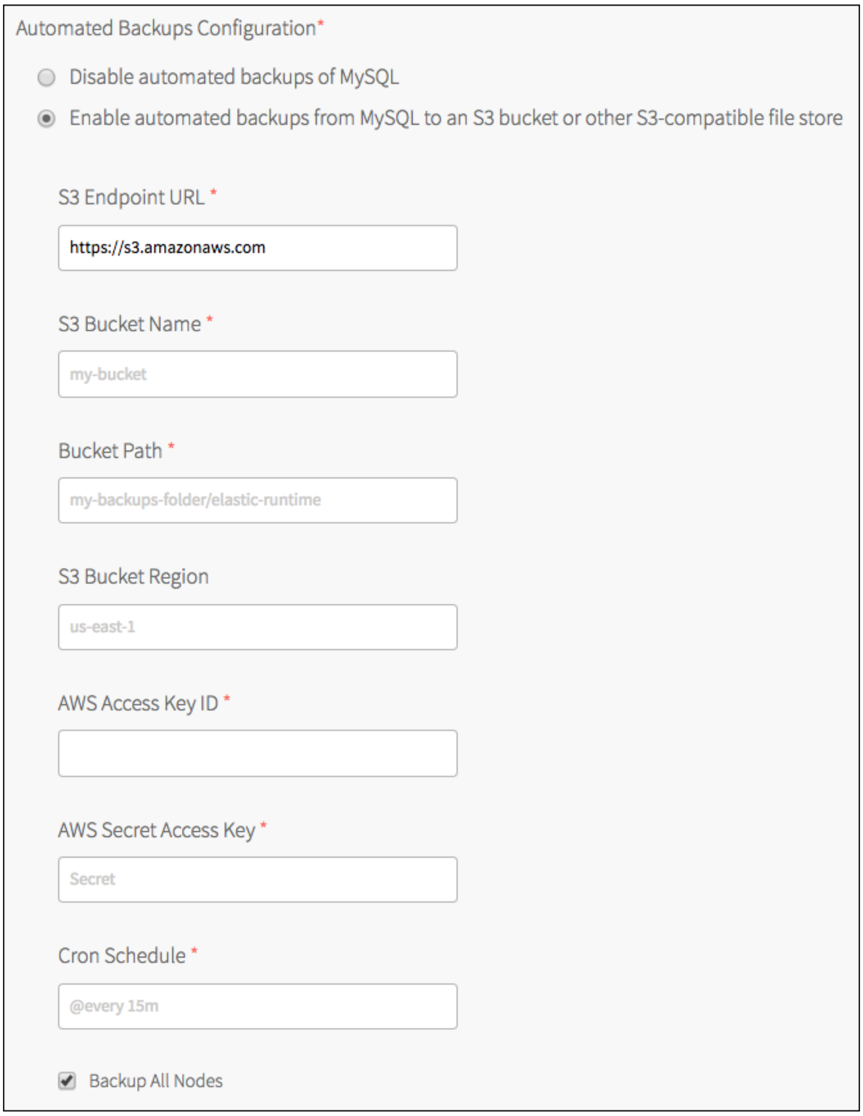
4. In the **Replication canary time period** field, leave the default of 30 seconds or modify the value based on the needs of your deployment. Lower numbers cause the canary to run more frequently, which means that the canary reacts more quickly to replication failure but adds load to the database.
5. In the **Replication canary read delay** field, leave the default of 20 seconds or modify the value based on the needs of your deployment. This field configures how long the canary waits, in seconds, before verifying that data is replicating across each MySQL node. Clusters under heavy load can experience a small replication lag as write-sets are committed across the nodes.
6. (**Required**): In the **E-mail address** field, enter the email address where the MySQL service sends alerts when the cluster experiences a replication issue or when a node is not allowed to auto-rejoin the cluster.
7. To prohibit the creation of command line history files on the MySQL nodes, disable the **Allow Command History** checkbox.
8. To allow the admin and roadmin to connect from any remote host, enable the **Allow Remote Admin Access** checkbox. When the checkbox is disabled, admins must `bosh ssh` into each MySQL VM to connect as the MySQL super user.

> **Note:** Network configuration and Application Security Groups restrictions may still limit a client's ability to establish a connection with the databases.

9. For **Cluster Probe Timeout**, enter the maximum amount of time, in seconds, that a new node will search for existing cluster nodes. If left blank, the default value is 10 seconds.



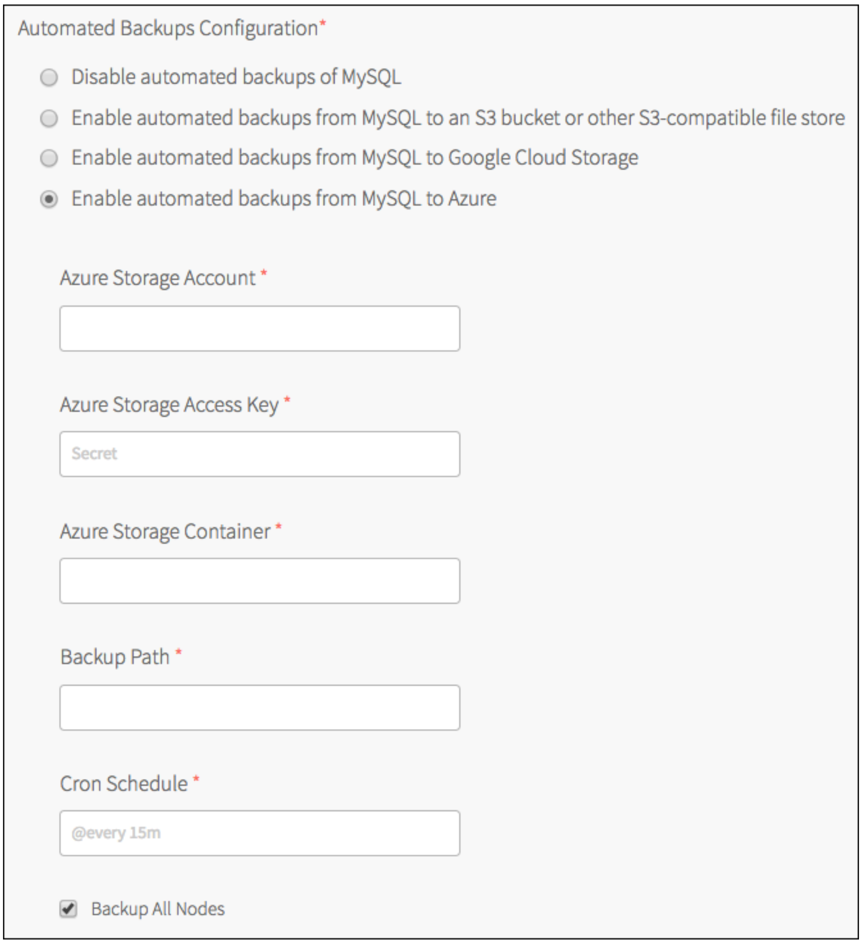
10. Under **Automated Backups Configuration**, choose one of three options for MySQL backups:
    - **Disable automatic backups of MySQL** disables automatic backups, but you can still deploy the Backup Prepare Node if you use BOSH Backup and Restore to back up your MySQL database. For more information, see the Backing Up Pivotal Cloud Foundry with BBR topic.
    - **Enable automated backups from MySQL to an S3 bucket or other S3-compatible file store** saves your backups to an existing Amazon Web Services (AWS) or Ceph S3-compatible blobstore.



This option requires the following fields:

- For **S3 Bucket Name**, enter the name of your S3 bucket. Do not include an `s3://` prefix, a trailing `/`, or underscores. If the bucket does not already exist, it will be created automatically.
- For **Bucket Path**, specify a folder within the bucket to hold your MySQL backups. Do not include a trailing `/`.
- For **S3 Bucket Region**, enter the AWS region where the bucket is located, such as `us-east-1`.
- For **AWS Access Key ID** and **AWS Secret Access Key**, enter your AWS or Ceph credentials.
- For **Cron Schedule**, enter a valid cron expression to schedule your automated backups. Cron uses your computer's local time zone.
- Enable **Backup All Nodes** to make unique backups from each instance of the MySQL server rather than just the first MySQL server instance.

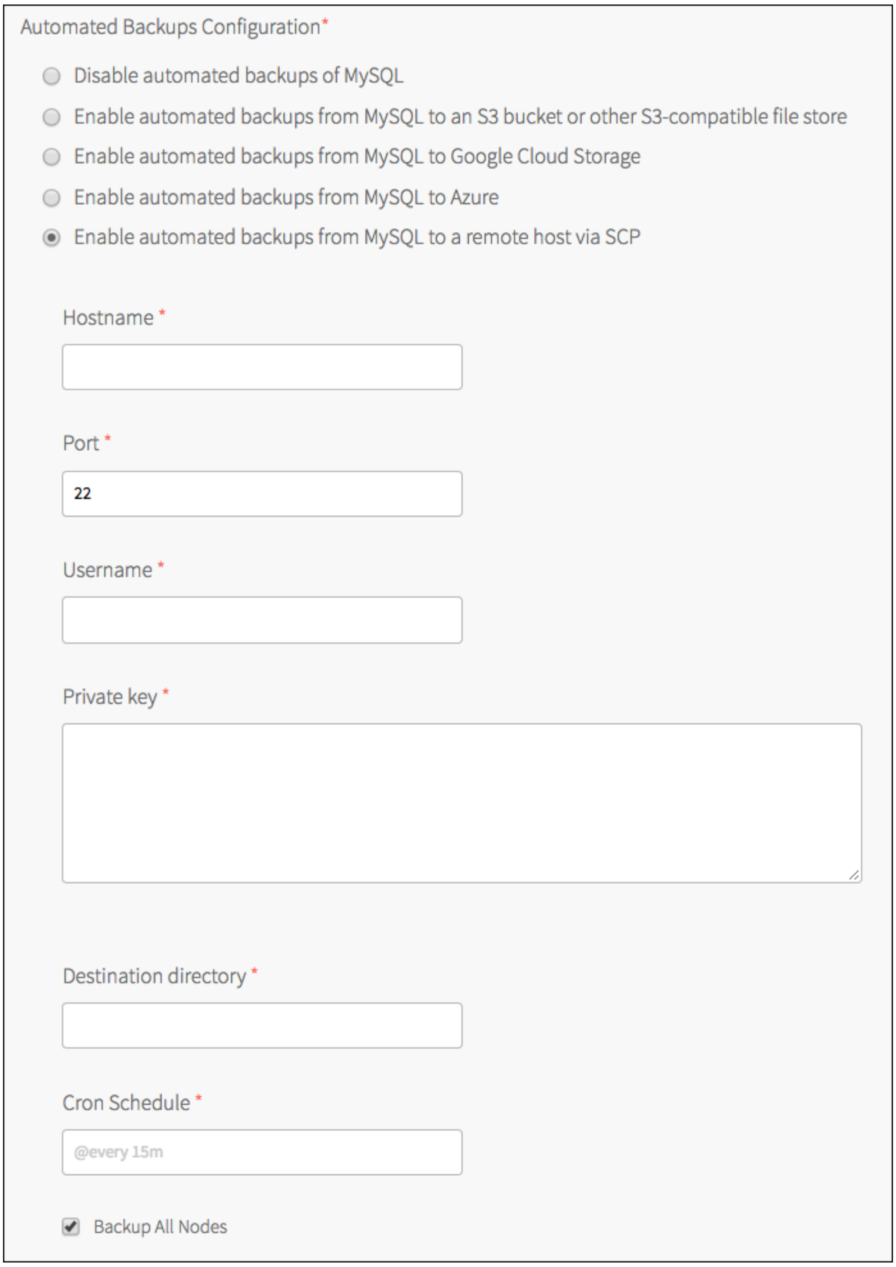
- **Enable automated backups from MySQL to Azure** saves your backups to Azure.



This option requires the following fields:

- For **Azure Storage Account**, enter the name of an existing Azure storage account where backups will be uploaded. For more information about creating and managing an Azure storage account, see the Azure documentation.
- For **Azure Storage Access Key**, enter an Azure storage access key for the storage account.
- For **Azure Storage Container**, enter the name of an existing Azure storage container that will store the backups.
- For **Backup Path**, enter the path within the Azure storage container where backups will be uploaded.
- For **Cron Schedule**, enter a valid cron expression to schedule your automated backups. Cron uses your computer's local time zone.
- Enable **Backup All Nodes** to make unique backups from each instance of the MySQL server rather than just the first MySQL server instance.

- **Enable automated backups from MySQL to a remote host via SCP**saves your backups to a remote host using secure copy protocol (SCP).



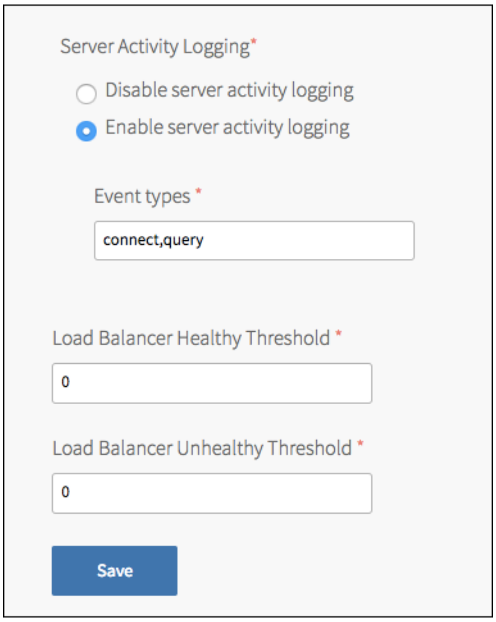
This option requires the following fields:

- For **Hostname**, enter the name of your SCP host.
- For **Port**, enter your SCP port. This should be the TCP port that your SCP host uses for SSH. The default port is `22`.
- For **Username**, enter your SSH username for the SCP host.
- For **Private key**, paste in your SSH private key.
- For **Destination directory**, enter the directory on the SCP host where you want to save backup files.
- For **Cron Schedule**, enter a valid cron expression to schedule your automated backups. Cron uses your computer's local time zone.
- Enable **Backup All Nodes** to make unique backups from each instance of the MySQL server rather than just the first MySQL server instance.

> **Note:** If you choose to enable automated MySQL backups, set the number of instances for the **Backup Prepare Node** under the **Resource Config** section of the Pivotal Application Service (PAS) tile to `1`.

11. If you want to log audit events for internal MySQL, select **Enable server activity logging** under **Server Activity Logging**.

    a. For the **Event types** field, you can enter the events you want the MySQL service to log. By default, this field includes `connect` and `query`, which tracks who connects to the system and what queries are processed. For more information, see the Logging Events section of the MariaDB documentation.



12. Enter values for the following fields:
    - **Load Balancer Healthy Threshold**: Specifies the amount of time, in seconds, to wait until declaring the MySQL proxy instance started. This allows an external load balancer time to register the instance as healthy.
    - **Load Balancer Unhealthy Threshold**: Specifies the amount of time, in seconds, that the MySQL proxy continues to accept connections before shutting down. During this period, the healthcheck reports as unhealthy to cause load balancers to fail over to other proxies. You must enter a value greater than or equal to the maximum time it takes your load balancer to consider a proxy instance unhealthy, given repeated failed healthchecks.

13. If you want to enable the MySQL interruptor feature, select the checkbox to **Prevent node auto re-join**. This feature stops all writes to the MySQL database if it notices an inconsistency in the dataset between the nodes. For more information, see the Interruptor section in the MySQL for PCF documentation.

14. Click **Save**.

Step 13: Configure File Storage

To minimize system downtime, Pivotal recommends using highly resilient and redundant*external* filestores for your Pivotal Application Service (PAS) file storage.

When configuring file storage for the Cloud Controller in PAS, you can select one of the following:

- Internal WebDAV filestore
- External S3-compatible or Ceph-compatible filestore
- External Google Cloud Storage
- External Azure Cloud Storage

For production-level PCF deployments on Azure, the recommended selection is Azure Storage. For more information about production-level PCF deployments on Azure, see the Reference Architecture for Pivotal Cloud Foundry on Azure.

For more factors to consider when selecting file storage, see Considerations for Selecting File Storage in Pivotal Cloud Foundry.

Internal Filestore

Internal file storage is only appropriate for small, non-production deployments.

To use the PCF internal filestore, perform the following steps:

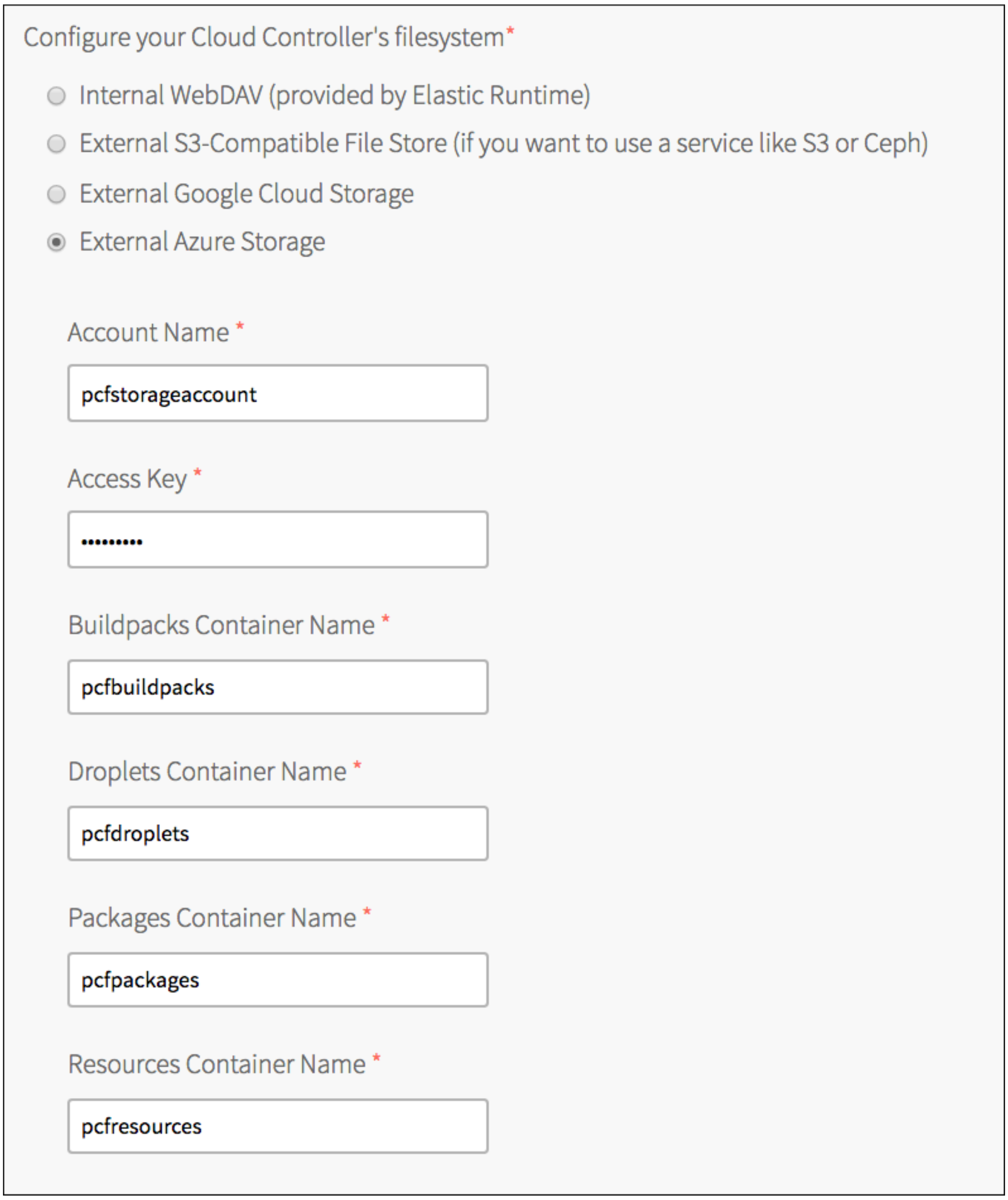
1. In the Pivotal Application Service (PAS) tile, select **File Storage**.

2. Select **Internal WebDAV**, and click **Save**.

External Azure Storage

To use external Azure file storage for your Pivotal Application Service (PAS) filestore, perform the following steps:

1. Select the **External AzureStorage** option.



2. To create a new storage account and storage containers for the PAS filestore, perform the following steps.
    - In the Azure Portal, navigate to the **Storage accounts** tab.
    - Click on the plus icon to add a new storage account.
    - In the **Name** field, enter a unique name (all lowercase, 3 to 24 alphanumeric characters) for the storage account.
    - For the **Deployment model**, select **Resource manager**.
    - For **Account kind**, select **General purpose**.
    - For **Performance**, select **Standard**.
    - From the **Replication** dropdown, select **Locally-redundant storage (LRS)**.
    - For **Storage service encryption**, select **Disabled**.
    - From the **Subscription** dropdown, select the subscription where you want to deploy PCF resources.
    - For **Resource group**, select **Use existing** and enter the name of the resource group where you deployed PAS.
    - From **Location** the dropdown, select the **Location** where you are deploying PCF.
    - Click **Create**.
    - After the storage account is created, select the new storage account from the dashboard.
    - Navigate to the **Blob Service** section of the storage account, and then click on **Containers** to create one or more containers in this storage account for buildpacks, droplets, resources, and packages.
    - For each container that you create, set the **Access type** to **Private**.

3. In PAS, enter the name of the storage account you created for **Account Name**.

4. In the **Secret Key** field, enter one of the access keys provided for the storage account. To obtain a value for this fields, visit the Azure Portal, navigate to the **Storage accounts** tab and click on **Access keys**.

5. For the **Buildpacks Container Name**, enter the container name for storing your app buildpacks.

6. For **Droplets Container Name**, enter the container name for your app droplet storage. Pivotal recommends that you use a unique container name, but you can use the same container name as the previous step.

7. For **Resources Container Name**, enter the container name for resources. Pivotal recommends that you use a unique container name, but you can use the same container name as the previous step.

8. For **Packages Container Name**, enter the container name for packages. Pivotal recommends that you use a unique container name, but you can use the same container name as the previous step.

9. Click **Save**.

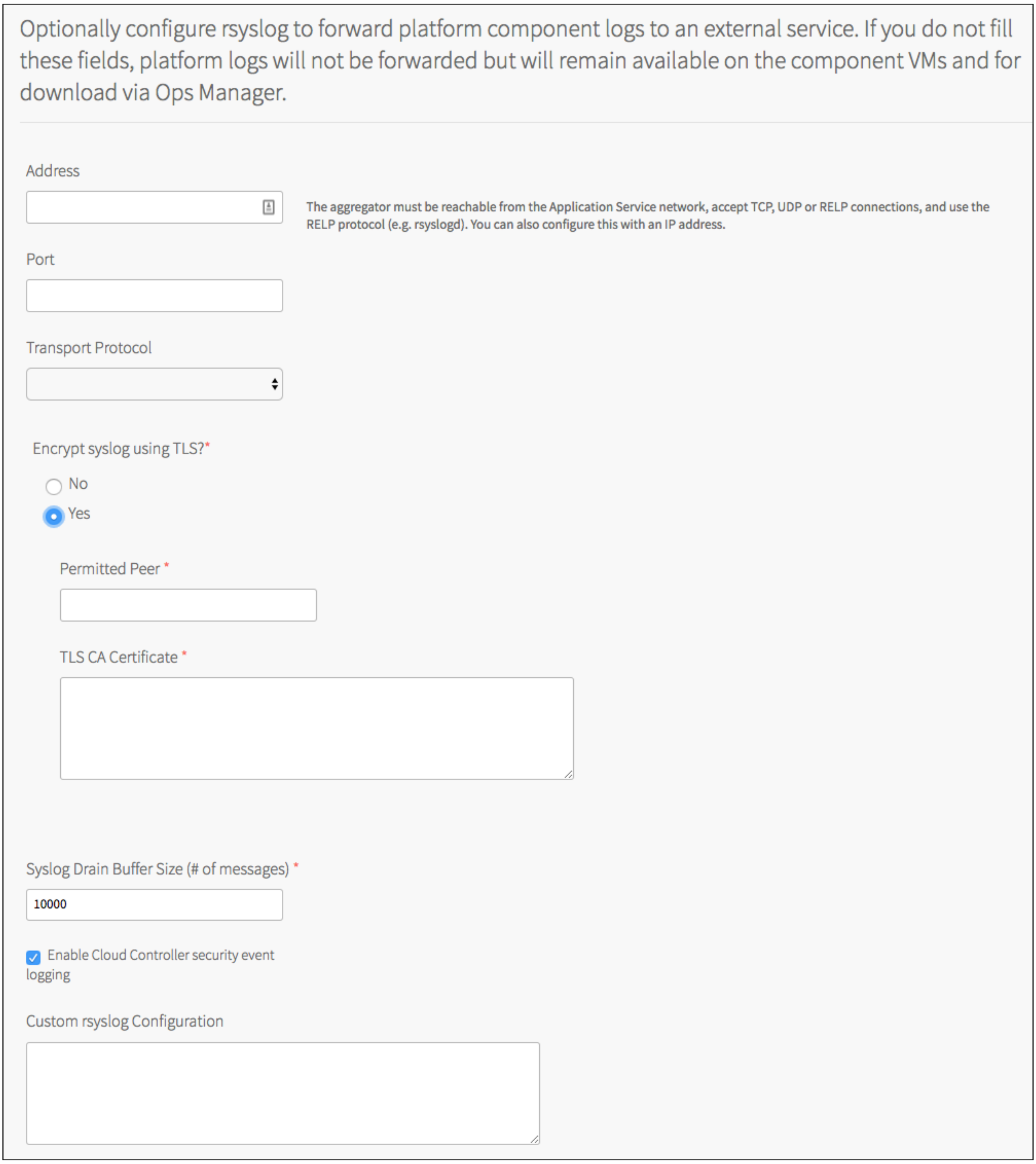
Other IaaS Storage Options

Google Cloud Storage and External S3-Compatible File Storage are also available as file storage options but are not recommended for typical PCF on Azure installations.

Step 14: (Optional) Configure System Logging

If you forward logging messages to an external Reliable Event Logging Protocol (RELP) server, complete the following steps:

1. Select the **System Logging** section that is located within your PAS **Settings** tab.



2. Enter the IP address of your syslog server in **Address**.

3. Enter the port of your syslog server in **Port**. The default port for a syslog server is `514`.

> **Note:** The host must be reachable from the PAS network, accept TCP connections, and use the RELP protocol. Ensure your syslog server listens on external interfaces.

4. Select a **Transport Protocol** to use when forwarding logs.

5. If you plan to use TLS encryption when sending logs to the remote server, select **Yes** when answering the **Encrypt syslog using TLS?** question.

   a. In the **Permitted Peer** field, enter either the name or SHA1 fingerprint of the remote peer.
   b. In the **TLS CA Certificate** field, enter the TLS CA Certificate for the remote server.

6. For the **Syslog Drain Buffer Size**, enter the number of messages the Doppler server can hold from Metron agents before the server starts to drop them. See the Loggregator Guide for Cloud Foundry Operators topic for more details.

7. If you want to include security events in your log stream, select the **Enable Cloud Controller security event logging** checkbox. This logs all API requests, including the endpoint, user, source IP address, and request result, in the Common Event Format (CEF).

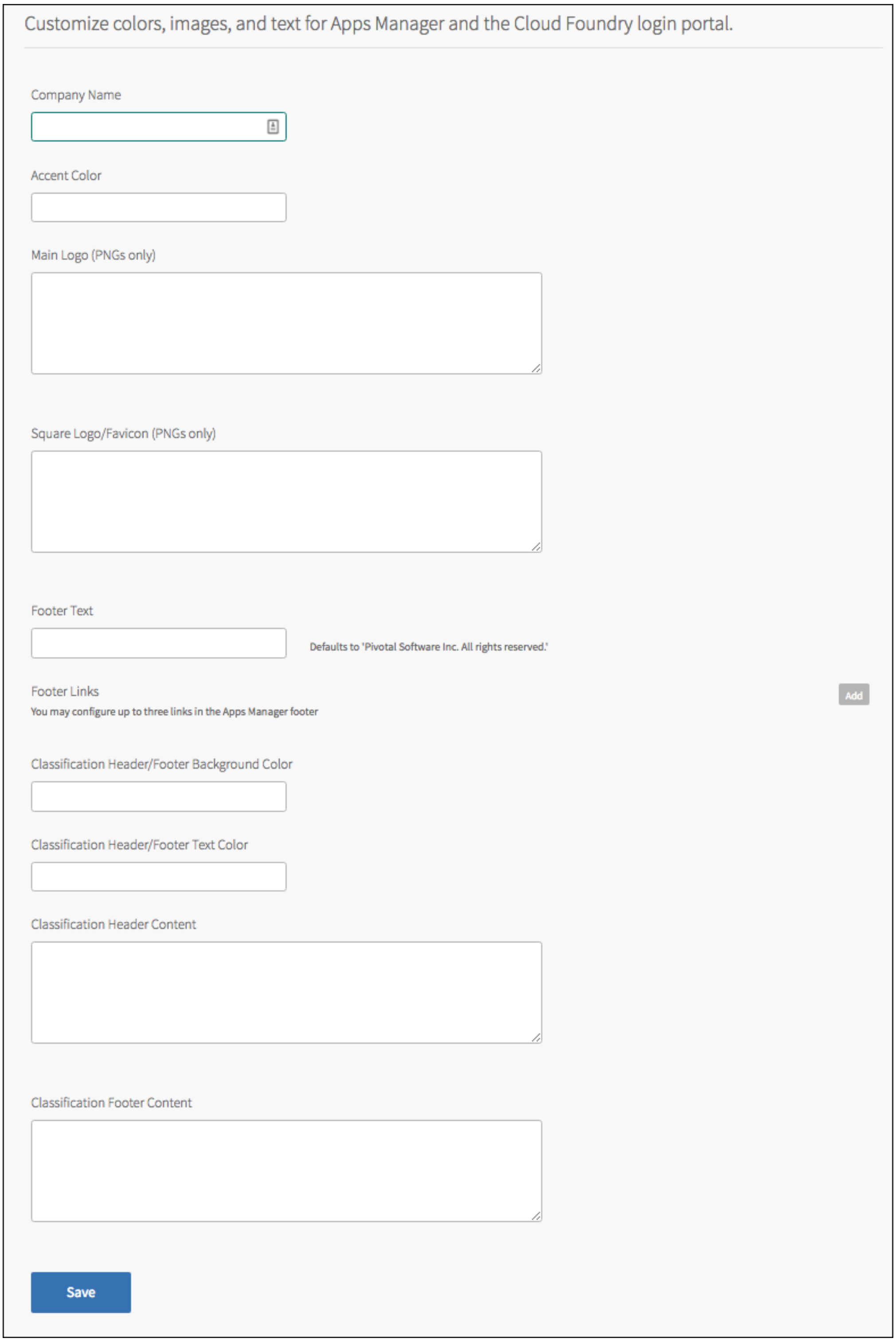
8. If you want to specify a custom syslog formatting rule, enter it in the **Custom rsyslog Configuration** field in RainerScript syntax.

9. Click **Save**.

Step 15: (Optional) Customize Apps Manager

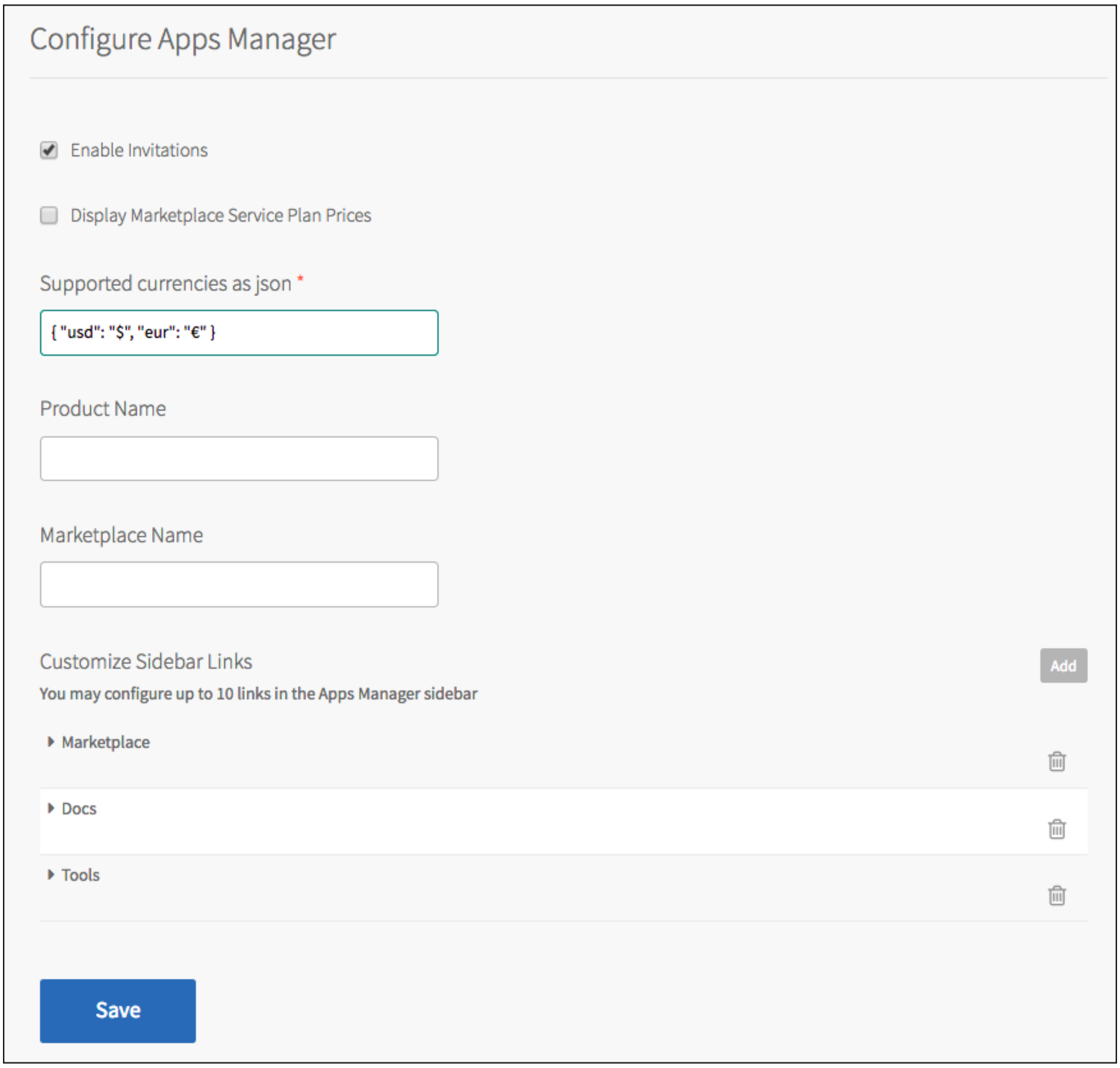
The **Custom Branding** and **Apps Manager** sections customize the appearance and functionality of Apps Manager. Refer to Custom Branding Apps Manager for descriptions of the fields on these pages and for more information about customizing Apps Manager.

1. Select **Custom Branding**. Use this section to configure the text, colors, and images of the interface that developers see when they log in, create an account, reset their password, or use Apps Manager.



2. Click **Save** to save your settings in this section.

3. Select **Apps Manager**.



4. Select **Enable Invitations** to enable invitations in Apps Manager. Space Managers can invite new users for a given space, Org Managers can invite new users for a given org, and Admins can invite new users across all orgs and spaces. See the Inviting New Users section of the *Managing User Roles with Apps Manager* topic for more information.

5. Select **Display Marketplace Service Plan Prices** to display the prices for your services plans in the Marketplace.

6. Enter the **Supported currencies as json** to appear in the Marketplace. Use the format `{"CURRENCY-CODE":"SYMBOL"}`. This defaults to `{"usd": "$", "eur": "€"}`.

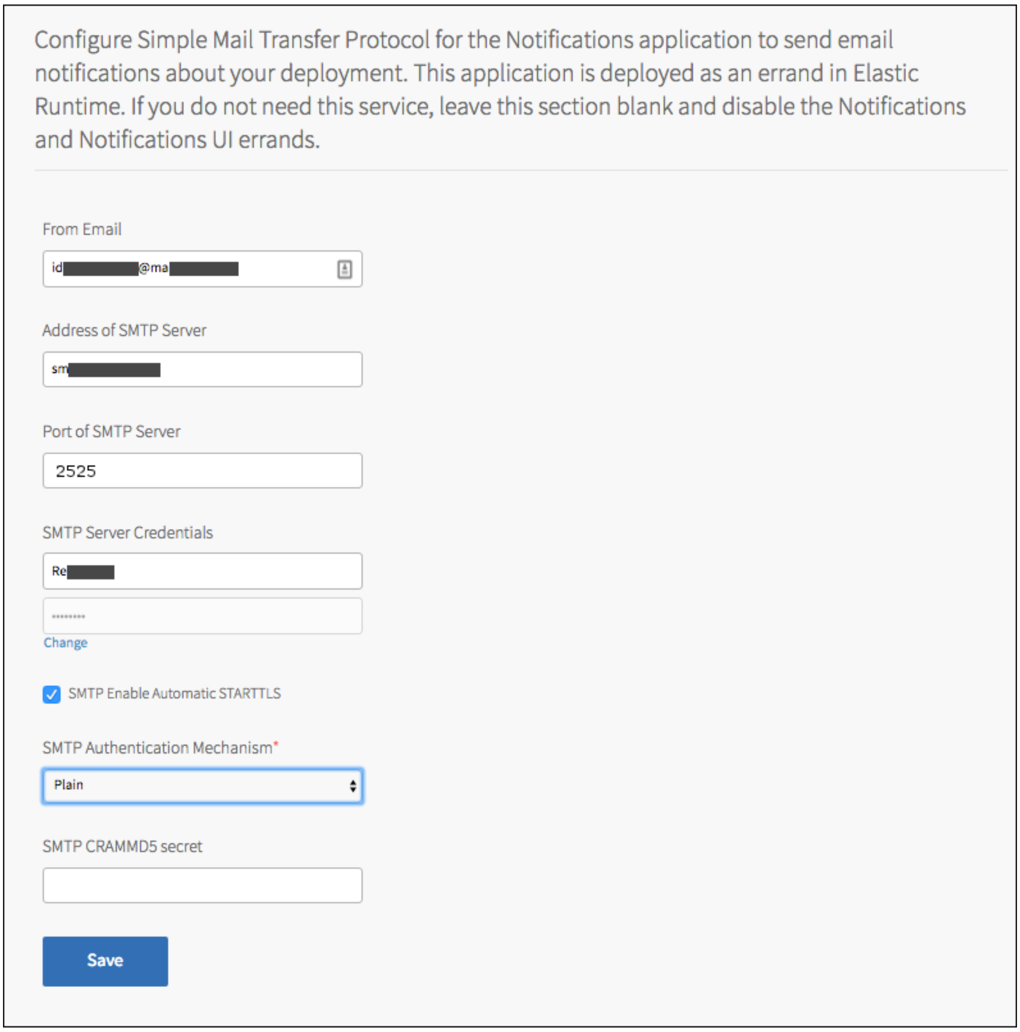
7. Use **Product Name**, **Marketplace Name**, and **Customize Sidebar Links** to configure page names and sidebar links in the **Apps Manager** and **Marketplace** pages.

8. Click **Save** to save your settings in this section.

Step 16: (Optional) Configure Email Notifications

PAS uses SMTP to send invitations and confirmations to Apps Manager users. You must complete the **Email Notifications** page if you want to enable end-user self-registration.

1. Select **Email Notifications**.

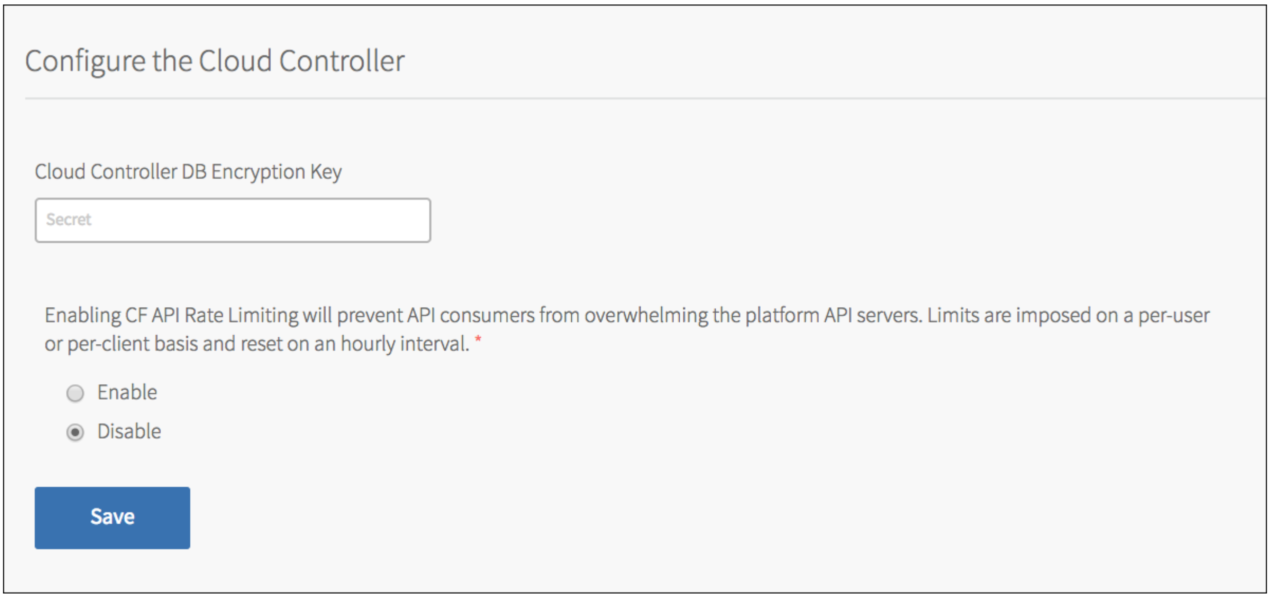


2. Enter your reply-to and SMTP email information
3. Verify your authentication requirements with your email administrator and use the **SMTP Authentication Mechanism** drop-down menu to select `None`, `Plain`, or `CRAMMD5`. If you have no SMTP authentication requirements, select `None`.
4. Click **Save**.

Note: If you do not configure the SMTP settings using this form, the administrator must create orgs and users using the cf CLI. See Creating and Managing Users with the cf CLI for more information.

Step 17: Configure Cloud Controller

1. Click **Cloud Controller**.



2. Enter your **Cloud Controller DB Encryption Key** if all of the following are true:
   - You deployed Pivotal Application Service (PAS) previously.
   - You then stopped PAS or it crashed.
   - You are re-deploying PAS with a backup of your Cloud Controller database.

     See [Backing Up Pivotal Cloud Foundry](#) for more information.

3. CF API Rate Limiting prevents API consumers from overwhelming the platform API servers. Limits are imposed on a per-user or per-client basis and reset on an hourly interval.

   To disable CF API Rate Limiting, select **Disable** under **Enable CF API Rate Limiting**. To enable CF API Rate Limiting, perform the following steps:

   a. Under **Enable CF API Rate Limiting**, select **Enable**.
   b. For **General Limit**, enter the number of requests a user or client is allowed to make over an hour interval for all endpoints that do not have a custom limit. The default value is `2000`.
   c. For **Unauthenticated Limit**, enter the number of requests an unauthenticated client is allowed to make over an hour interval. The default value is `100`.

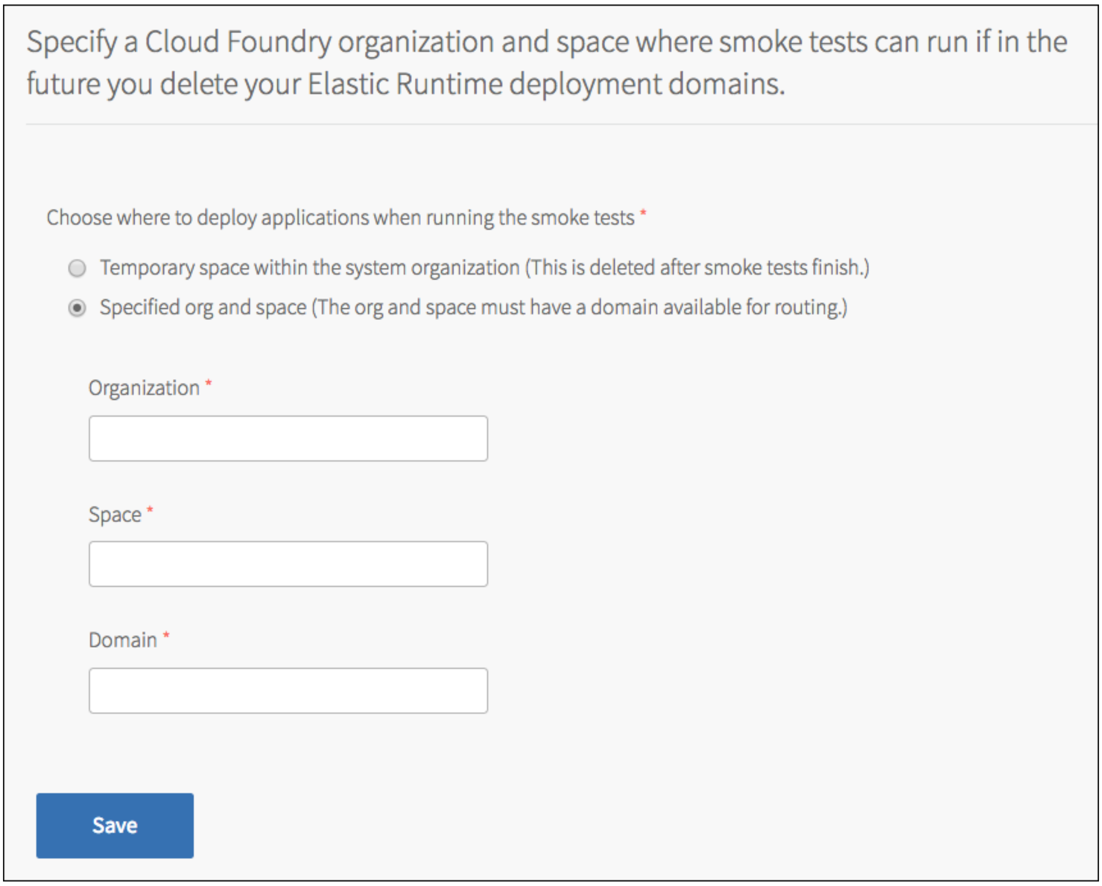
4. Click **Save**.

Step 18: Configure Smoke Tests

The Smoke Tests errand runs basic functionality tests against your Pivotal Application Service (PAS) deployment after an installation or update. In this section, choose where to run smoke tests. In the **Errands** section, you can choose whether or not to run the Smoke Tests errand.

1. Select **Smoke Tests**.

2. If you have a shared apps domain, select **Temporary space within the system organization**, which creates a temporary space within the `system` organization for running smoke tests and deletes the space afterwards. Otherwise, select **Specified org and space** and complete the fields to specify where you want to run smoke tests.



3. Click **Save**.

Step 19: (Optional) Enable Advanced Features

The **Advanced Features** section of Pivotal Application Service (PAS) includes new functionality that may have certain constraints. Although these features are fully supported, Pivotal recommends caution when using them in production environments.

Diego Cell Memory and Disk Overcommit

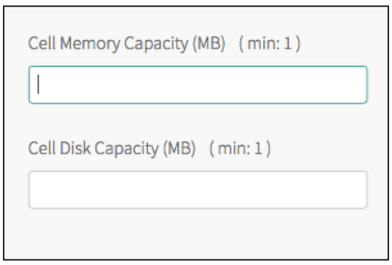
If your apps do not use the full allocation of disk space and memory set in the **Resource Config** tab, you might want use this feature. These fields control the amount to overcommit disk and memory resources to each Diego Cell VM.

For example, you might want to use the overcommit if your apps use a small amount of disk and memory capacity compared to the amounts set in the **Resource Config** settings for **Diego Cell**.

> **Note**: Due to the risk of app failure and the deployment-specific nature of disk and memory use, Pivotal has no recommendation about how much, if any, memory or disk space to overcommit.

To enable overcommit, follow these steps:

1. Select **Advanced Features**.



2. Enter the total desired amount of Diego cell memory value in the **Cell Memory Capacity (MB)** field. Refer to the **Diego Cell** row in the **Resource Config** tab for the current Cell memory capacity settings that this field overrides.

3. Enter the total desired amount of Diego cell disk capacity value in the **Cell Disk Capacity (MB)** field. Refer to the **Diego Cell** row in the **Resource Config** tab for the current Cell disk capacity settings that this field overrides.

4. Click **Save**.

Note: Entries made to each of these two fields set the total amount of resources allocated, not the overage.

Whitelist for Non-RFC-1918 Private Networks

Some private networks require extra configuration so that internal file storage (WebDAV) can communicate with other PCF processes.

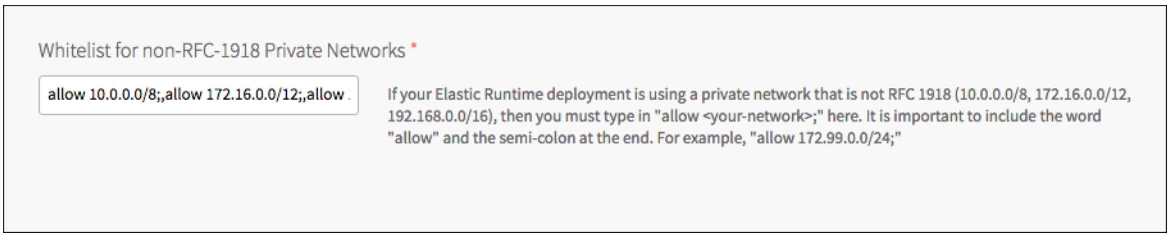
The **Whitelist for non-RFC-1918 Private Networks** field is provided for deployments that use a non-RFC 1918 private network. This is typically a private network other than `10.0.0.0/8`, `172.16.0.0/12`, or `192.168.0.0/16`.

Most PCF deployments do not require any modifications to this field.

To add your private network to the whitelist, perform the following steps:

1. Select **Advanced Features**.

2. Append a new `allow` rule to the existing contents of the **Whitelist for non-RFC-1918 Private Networks** field.



Include the word `allow`, the network CIDR range to allow, and a semi-colon (`;`) at the end. For example: `allow 172.99.0.0/24;`

3. Click **Save**.

CF CLI Connection Timeout

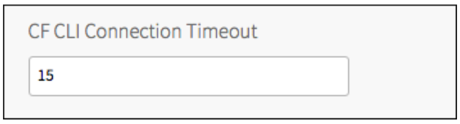
The **CF CLI Connection Timeout** field allows you to override the default five second timeout of the Cloud Foundry Command Line Interface (cf CLI) used within your PCF deployment. This timeout affects the cf CLI command used to push PAS errand apps such as Notifications, Autoscaler, and Apps Manager.

Set the value of this field to a higher value, in seconds, if you are experiencing domain name resolution timeouts when pushing errands in PAS.

To modify the value of the **CF CLI Connection Timeout**, perform the following steps:

1. Select **Advanced Features**.

2. Add a value, in seconds, to the **CF CLI Connection Timeout** field.



3. Click **Save**.

Step 20: Configure Errands

Errands are scripts that Ops Manager runs automatically when it installs or uninstalls a product, such as a new version of Pivotal Application Service (PAS). There are two types of errands: *post-deploy errands* run after the product is installed, and *pre-delete errands* run before the product in uninstalled.

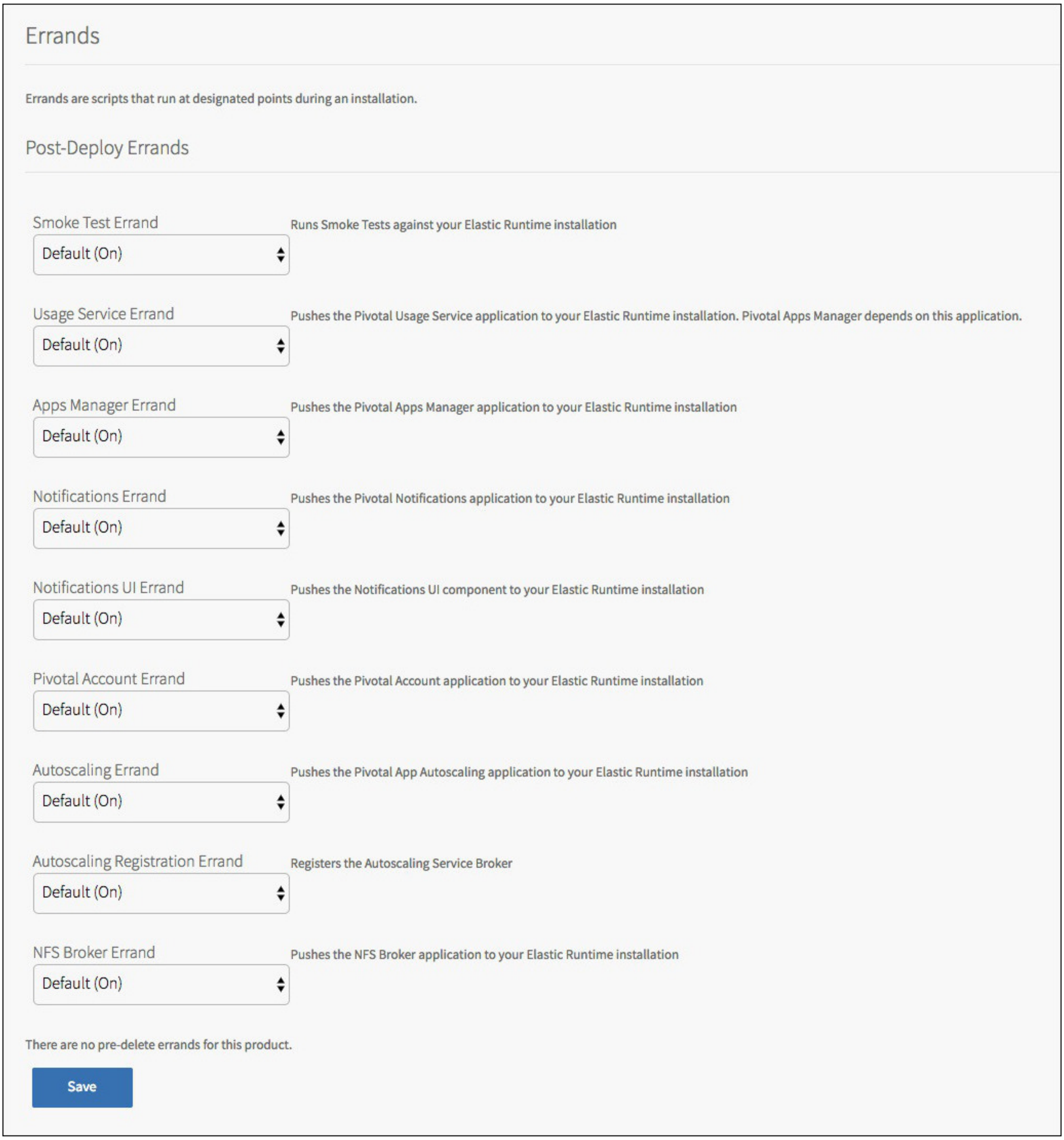
By default, Ops Manager always runs pre-delete errands, and only runs post-deploy errands when the product has changed since the last time Ops

Manager installed something. In PAS, the Smoke Test Errand defaults to always run.

The PAS tile **Errands** pane lets you change these run rules. For each errand, you can select **On** to run it always, **Off** to never run it, or **When Changed** to run it only when the product has changed since the last install.

For more information about how Ops Manager manages errands, see the Managing Errands in Ops Manager topic.

> **Note**: Several errands deploy apps that provide services for your deployment, such as Autoscaling and Notifications. Once one of these apps is running, selecting **Off** for the corresponding errand on a subsequent installation does not stop the app.



- **Smoke Test Errand** verifies that your deployment can do the following:
  - Push, scale, and delete apps
  - Create and delete orgs and spaces
- **Usage Service Errand** deploys the Pivotal Usage Service application, which Apps Manager depends on.
- **Apps Manager Errand** deploys Apps Manager, a dashboard for managing apps, services, orgs, users, and spaces. Until you deploy Apps Manager, you

must perform these functions through the cf CLI. After Apps Manager has been deployed, Pivotal recommends deselecting the checkbox for this errand on subsequent PAS deployments. For more information about Apps Manager, see the Getting Started with the Apps Manager topic.

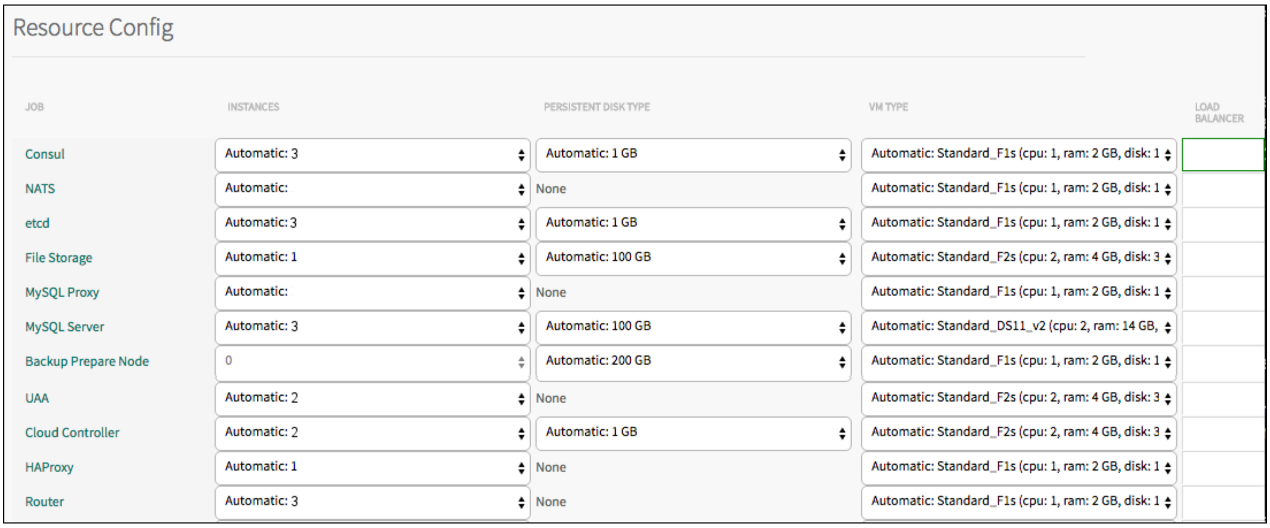
- **Notifications Errand** deploys an API for sending email notifications to your PCF platform users.

> **Note**: The Notifications app requires that you configure SMTP with a username and password, even if you set the value of **SMTP Authentication Mechanism** to `none`.

- **Notifications UI Errand** deploys a dashboard for users to manage notification subscriptions.
- **Pivotal Account Errand** deploys Pivotal Account, a dashboard that allows users to create and manage their accounts. In the Pivotal Account dashboard, users can launch applications, manage their profiles, manage account security, manage notifications, and manage approvals. See the Enabling Pivotal Account topic for more information.
- **Autoscaling Errand** enables you to configure your apps to automatically scale in response to changes in their usage load. See the Scaling an Application Using Autoscaler topic for more information.
- **Autoscaling Registration Errand** makes the Autoscaling service available to your applications. Without this errand, you cannot bind the Autoscaling app to your apps.
- **NFS Broker Errand** enables you to use NFS Volume Services by installing the NFS Broker app in PAS. See the Enabling NFS Volume Services topic for more information.

Step 21: Configure Resources

1. Select **Resource Config**.



Resource Config

JOB	INSTANCES	PERSISTENT DISK TYPE	VM TYPE	LOAD BALANCER
Consul	Automatic: 3	Automatic: 1 GB	Automatic: Standard_F1s (cpu: 1, ram: 2 GB, disk: 1	
NATS	Automatic:	None	Automatic: Standard_F1s (cpu: 1, ram: 2 GB, disk: 1	
etcd	Automatic: 3	Automatic: 1 GB	Automatic: Standard_F1s (cpu: 1, ram: 2 GB, disk: 1	
File Storage	Automatic: 1	Automatic: 100 GB	Automatic: Standard_F2s (cpu: 2, ram: 4 GB, disk: 3	
MySQL Proxy	Automatic:	None	Automatic: Standard_F1s (cpu: 1, ram: 2 GB, disk: 1	
MySQL Server	Automatic: 3	Automatic: 100 GB	Automatic: Standard_DS11_v2 (cpu: 2, ram: 14 GB,	
Backup Prepare Node	0	Automatic: 200 GB	Automatic: Standard_F1s (cpu: 1, ram: 2 GB, disk: 1	
UAA	Automatic: 2	None	Automatic: Standard_F2s (cpu: 2, ram: 4 GB, disk: 3	
Cloud Controller	Automatic: 2	Automatic: 1 GB	Automatic: Standard_F2s (cpu: 2, ram: 4 GB, disk: 3	
HAProxy	Automatic: 1	None	Automatic: Standard_F1s (cpu: 1, ram: 2 GB, disk: 1	
Router	Automatic: 3	None	Automatic: Standard_F1s (cpu: 1, ram: 2 GB, disk: 1	

2. Ensure a `Standard` VM type is selected for the **Router** VM. The PAS deployment fails if you select a `Basic` VM type.

3. Retrieve the name(s) of your external ALB by navigating to the Azure portal, clicking **All resources**, and locating your **Load balancer** resource. If you used the ARM Template to launch Ops Manager Director, then the name of the load balancer should be `ERT-LB`.

> **Note**: The Azure portal sometimes displays the names of resources with incorrect capitalization. Always use the Azure CLI to retrieve the correctly capitalized name of a resource.

4. Locate the **HAProxy** job in the **Resource Config** pane and enter the name of your external ALB in the field under **Load Balancers**.

> **Note**: Do not enter a load balancer for the **Diego Brain** component.

5. Ensure that the **Internet Connected** checkboxes are deselected for all jobs.

6. Scale the number of instances as appropriate for your deployment.

> **Note**: For a high availability deployment of PCF on Azure, Pivotal recommends scaling the number of each PAS job to a minimum of three (3) instances. Using three or more instances for each job creates a sufficient number of availability sets and fault domains for your deployment. For more information, see Reference Architecture for Pivotal Cloud Foundry on Azure.

Step 22: (Optional) Scale Down and Disable Resources

> **Note**: The **Resource Config** pane has fewer VMs if you are installing the Small Footprint Runtime.

> **Note**: The Small Footprint Runtime does not default to a highly available configuration. It defaults to the minimum configuration. If you want to make the Small Footprint Runtime highly available, scale the **Compute**, **Router**, and **Database** VMs to `3` instances and scale the **Control** VM to `2` instances.

Pivotal Application Service (PAS) defaults to a highly available resource configuration. However, you may need to perform additional procedures to make your deployment highly available. See the Zero Downtime Deployment and Scaling in CF and the Scaling Instances in PAS topics for more information.

If you do not want a highly available resource configuration, you must scale down your instances manually by using the drop-down menus under **Instances** for each job.

By default, PAS also uses an internal filestore and internal databases. If you configure PAS to use external resources, you can disable the corresponding system-provided resources in Ops Manager to reduce costs and administrative overhead.

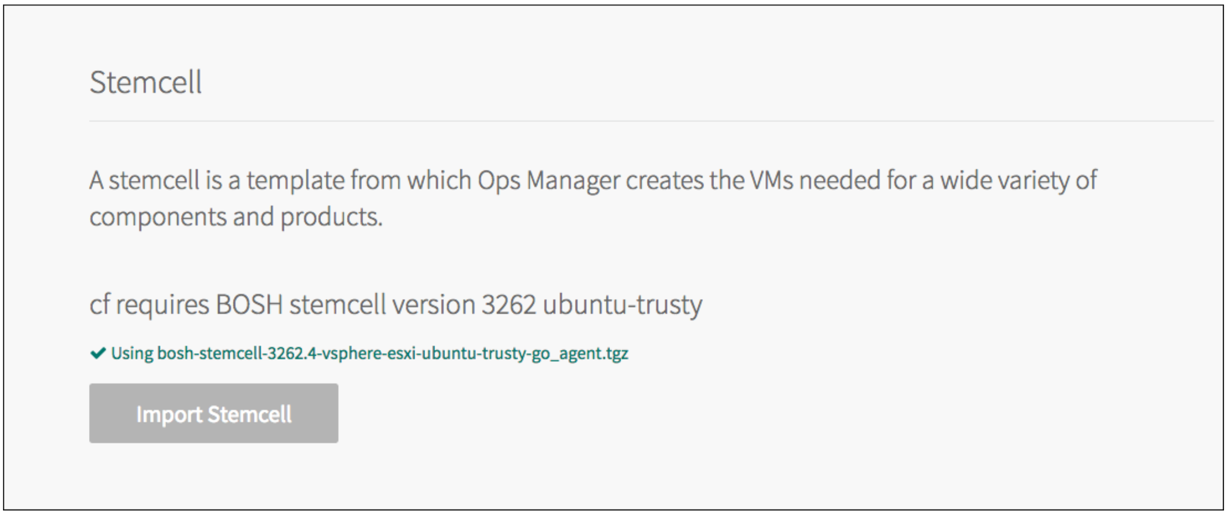
Complete the following procedures to disable specific VMs in Ops Manager:

1. Click **Resource Config**.
2. If you configure PAS to use an external S3-compatible filestore, edit the following fields:
   - **File Storage**: Enter `0` in **Instances**.
3. If you selected **External** when configuring the UAA or System databases, edit the following fields:
   - **MySQL Proxy**: Enter `0` in **Instances**.
   - **MySQL Server**: Enter `0` in **Instances**.
   - **MySQL Monitor**: Enter `0` in **Instances**.
   - **Cloud Controller Database**: Enter `0` in **Instances**.
   - **UAA Database**: Enter `0` in **Instances**.
4. Click **Save**.

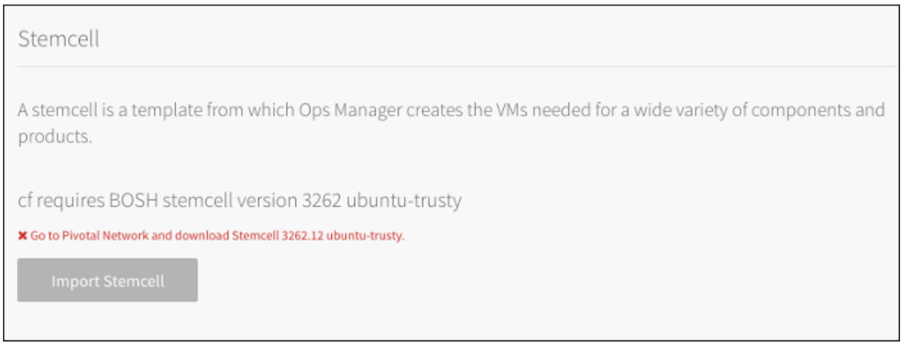
Step 23: Configure Stemcell

Verify whether Ops Manager is providing the stemcell version required by PAS. If the correct version is already present, you do not need to download a new stemcell.

1. In the PAS tile, select **Stemcell**.
2. Verify that the version indicated in the filename matches the version of stemcell required by PAS.
   - If PAS detects that a stemcell `.tgz` file is present in the Ops Manager Director VM at `/var/tempest/stemcells/`, the Stemcell screen displays filename information.



   - If PAS cannot detect a stemcell `.tgz` file, the following message displays:



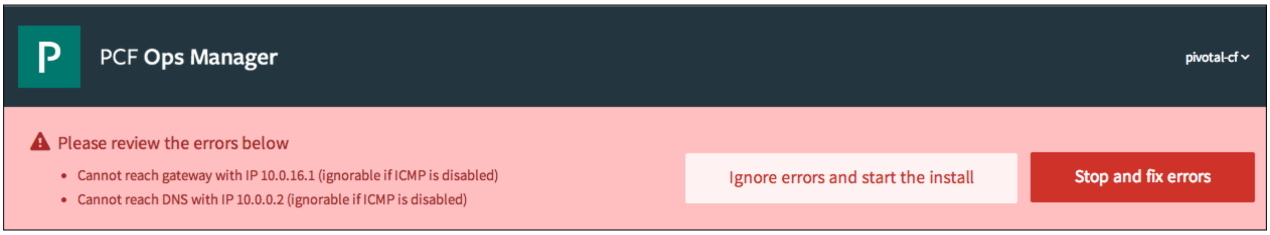
3. If the version of the stemcell file that is loaded does not match the required version listed in the Pivotal Network download page for PAS, or cannot be found by Ops Manager, perform the following steps to download and import a new stemcell file:

    a. Log in to the Pivotal Network and click **Stemcells**.
    b. Download the appropriate stemcell version targeted for your IaaS.
    c. In the **Stemcell** section of the PAS tile, click **Import Stemcell** to import the downloaded stemcell `.tgz` file.

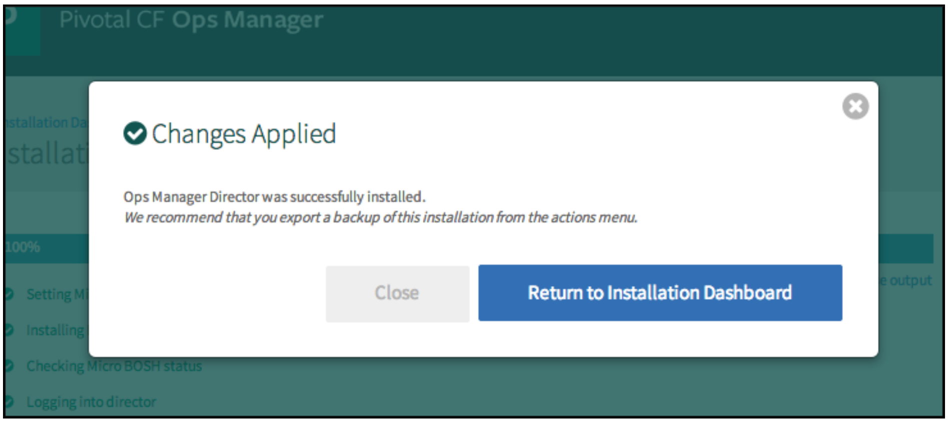
Step 24: Complete the PAS Installation

1. Click the **Installation Dashboard** link to return to the Installation Dashboard.

2. Click **Apply Changes**. If the following ICMP error message appears, click **Ignore errors and start the install**.



The install process generally requires a minimum of 90 minutes to complete. The image shows the Changes Applied window that displays when the installation process successfully completes.



Pivotal

Configuring Ops Manager on Azure Government Cloud

Page last updated:

This topic describes how to configure Pivotal Cloud Foundry (PCF) Ops Manager on Azure Government Cloud.

Before performing this procedure, you must complete the steps in Prepare Azure and Deploying BOSH and Ops Manager to Azure Manually.

The procedures below involve using the Ops Manager API. For more information about the Ops Manager API, see the documentation at `https://OPS-MAN-FQDN/docs`.

Step 1: Configure Authentication

Perform the procedures in the Step 1: Access Ops Manager section of the *Configuring Ops Manager Director on Azure* topic. Set up your Ops Manager authentication system, but do not continue to the Step 2: Azure Config Page section.

Continue to the Internal Authentication or External Identity Provider section below depending on which authentication system you configured.

Internal Authentication

If you configured your Ops Manager for Internal Authentication, perform the following steps:

1. SSH into the Ops Manager VM:

   ```
   $ ssh -i opsman ubuntu@OPS_MAN_FQDN
   ```

   If the private key that you generated in the Launching an Ops Manager Director Instance on Azure without an ARM Template topic is not named `opsman`, provide the correct filename instead.

2. From the Ops Manager VM, use the User Account and Authentication Command Line Interface (UAAC) to target your Ops Manager UAA server:

   ```
   $ uaac target https://OPS-MAN-FQDN/uaa
   ```

   Note: UAA is the Cloud Foundry identity and management service. See the User Account and Authentication (UAA) Server topic for more information.

3. Retrieve your token to authenticate:

   ```
   $ uaac token owner get
   Client ID: opsman
   Client secret: [Leave Blank]
   User name: OPS-MAN-USERNAME
   Password: OPS-MAN-PASSWORD
   ```

4. Continue to Step 2: Configure Ops Manager.

External Identity Provider

If you configured your Ops Manager for an external Identity Provider with SAML, perform the following steps:

1. From your local machine, target your Ops Manager UAA server:

   ```
   $ uaac target https://OPS-MAN-FQDN/uaa
   ```

2. Retrieve your token to authenticate. When prompted for a passcode, retrieve it from `https://OPS-MAN-FQDN/uaa/passcode`.

```
$ uaac token sso get
Client ID: opsman
Client secret: [Leave Blank]
Passcode: YOUR-PASSCODE
```

If authentication is successful, the UAAC displays the following message: `Successfully fetched token via owner password grant.`

3. Continue to Step 2: Configure Ops Manager.

Step 2: Configure Ops Manager

Perform the following steps to configure Ops Manager for Azure Government Cloud:

1. List your tokens.

```
$ uaac contexts
```

Locate the entry for your Ops Manager FQDN. Under `client_id: opsman`, record the value for `access_token`.

2. Use `curl` to pass in the `environment:AzureUSGovernment` key to Ops Manager using the API:

```
$ curl "https://OPS-MAN-FQDN/api/v0/staged/director/properties" \
 -X PUT \
 -H "Authorization: Bearer UAA-ACCESS-TOKEN" \
 -H "Content-Type: application/json" \
 -d '{
  "iaas_configuration": {
   "subscription_id": "SUBSCRIPTION-ID",
   "tenant_id": "TENANT-ID",
   "client_id": "APPLICATION-ID",
   "resource_group_name": "$RESOURCE-GROUP",
   "bosh_storage_account_name": "$STORAGE-NAME",
   "deployments_storage_account_name": "MY-DEPLOYMENT-STORAGE-X",
   "default_security_group": "opsmgr-nsg",
   "ssh_public_key": "ssh-rsa OPS-MAN-PUBLIC-KEY",
   "ssh_private_key": "-----BEGIN EXAMPLE RSA PRIVATE KEY-----\OPS-MAN-PRIVATE-KEY",
   "environment": "AzureUSGovernment"
  },
  "director_configuration": {
   "ntp_servers_string": "us.pool.ntp.org",
   "metrics_ip": "1.2.3.4",
   "resurrector_enabled": true,
   "max_threads": 1,
   "database_type": "internal",
   "blobstore_type": "local"
  },
  "security_configuration": {
  "trusted_certificates": "—— BEGIN SSL CERTIFICATE —— ... ",
  "generate_vm_passwords": true
 }
}'
```

Replace the placeholder values under `iaas_configuration` as follows:

- `UAA_ACCESS_TOKEN` is the token you retrieved from `uaac contexts` in the previous step.
- `SUBSCRIPTION_ID`, `TENANT_ID`, and `APPLICATION_ID` are set in the Preparing to Deploy PCF on Azure topic.
- The `$RESOURCE_GROUP` and `$STORAGE_NAME` environment variables are set in the Launching an Ops Manager Director Instance on Azure without an ARM Template topic.
- For the values for `MY_DEPLOYMENT_STORAGE_X`, `OPS_MAN_PUBLIC_KEY`, and `OPS_MAN_PRIVATE_KEY`, see the procedures in the Launching an Ops Manager Director Instance on Azure without an ARM Template topic.

Leave the values under `director_configuration` and `security_configuration`. They are designed to be overridden in a subsequent step when you configure Ops Manager using the web interface.

3. If the `curl` command returns a `200 OK` response, navigate to the Ops Manager FQDN in a browser and log in.

4. Configure the Ops Manager Director tile by performing the procedures in the Step 2: Azure Config Page section through the Step 6: Security Page section of the Configuring Ops Manager Director on Azure topic. Stop when you reach the **Resource Config** page in Ops Manager.

5. Only certain VM types are compatible with Azure Government Cloud. Create a file called `vmtypes` with the following contents: `{"vm_types": [{ "name": "Standard_D1_v2", "ram": 3584, "cpu": 1, "ephemeral_disk": 51200 }, { "name": "Standard_D2_v2", "ram": 7168, "cpu": 2, "ephemeral_disk": 102400 }, { "name": "Standard_D3_v2", "ram": 14336, "cpu": 4, "ephemeral_disk": 204800 }, { "name": "Standard_D4_v2", "ram": 28672, "cpu": 8, "ephemeral_disk": 409600 }, { "name": "Standard_D5_v2", "ram": 57344, "cpu":`

8, "ephemeral_disk": 819200 }, { "name": "Standard_D11_v2", "ram": 14336, "cpu": 2, "ephemeral_disk": 102400 }, { "name": "Standard_D12_v2", "ram": 28672, "cpu": 4, "ephemeral_disk": 204800 }, { "name": "Standard_D13_v2", "ram": 57344, "cpu": 8, "ephemeral_disk": 409600 }, { "name": "Standard_D14_v2", "ram": 114688, "cpu": 16, "ephemeral_disk": 819200 }, { "name": "Standard_F1", "ram": 2048, "cpu": 1, "ephemeral_disk": 16384 }, { "name": "Standard_F2", "ram": 4096, "cpu": 2, "ephemeral_disk": 32768 }, { "name": "Standard_F4", "ram": 8192, "cpu": 4, "ephemeral_disk": 65536 }, { "name": "Standard_F8", "ram": 16384, "cpu": 8, "ephemeral_disk": 131072]}

6. Use `curl` to pass the file with the correct VM types to Ops Manager using the API:

```
$ curl "https://OPS-MAN-FQDN/api/v0/vm_types" \
  -X PUT \
  -H "Authorization: Bearer UAA-ACCESS-TOKEN" \
  -H "Content-Type: application/json" \
  -d @vmtypes
```

7. If the `curl` command returns a `200 OK` response, retrieve the current list of VM types to ensure they are the ones you specified in the `vmtypes` file:

```
$ curl "https://OPS-MAN-FQDN/api/v0/vm_types" \
  -H "Authorization: Bearer UAA-ACCESS-TOKEN"

  200 OK
  RESPONSE:
  "vm_types": [ { "name": "Standard_D1_v2", "ram":
  3584, "cpu": 1, "ephemeral_disk": 51200 }, {
  "name": "Standard_D2_v2", "ram": 7168, "cpu": 2,
  "ephemeral_disk": 102400 }, { "name":
  "Standard_D3_v2", "ram": 14336, "cpu": 4,
  "ephemeral_disk": 204800 },{ "name":
  "Standard_D4_v2", "ram": 28672, "cpu": 8,
  "ephemeral_disk": 409600 },{ "name":
  "Standard_D5_v2", "ram": 57344, "cpu": 8,
  "ephemeral_disk": 819200 },{ "name":
  "Standard_D11_v2", "ram": 14336, "cpu": 2,
  "ephemeral_disk": 102400 },{ "name":
  "Standard_D12_v2", "ram": 28672, "cpu": 4,
  "ephemeral_disk": 204800 },{ "name":
  "Standard_D13_v2", "ram": 57344, "cpu": 8,
  "ephemeral_disk": 409600 },{ "name":
  "Standard_D14_v2", "ram": 114688, "cpu": 16,
  "ephemeral_disk": 819200 },{ "name": "Standard_F1",
  "ram": 2048, "cpu": 1, "ephemeral_disk": 16384 },
  { "name": "Standard_F2", "ram": 4096, "cpu": 2,
  "ephemeral_disk": 32768 }, {"name": "Standard_F4",
  "ram": 8192, "cpu": 4, "ephemeral_disk": 65536 },
  { "name": "Standard_F8", "ram": 16384, "cpu": 8,
  "ephemeral_disk": 131072 ]
```

8. Return to the Ops Manager Installation Dashboard by navigating to the Ops Manager FQDN in a browser.

9. Click **Apply Changes** to redeploy Ops Manager.

10. When Ops Manager finishes deploying, you can optionally deploy BOSH Add-Ons to your system, and then install one or more PCF runtime environments to complete your PCF installation.

Configuring Ops Manager on Azure Germany

Page last updated:

This topic describes how to configure Pivotal Cloud Foundry (PCF) Ops Manager on Azure Germany.

> **Note**: Azure Germany is only supported in PCF 1.10.1 and later.

Before performing this procedure, you must complete the steps in Prepare Azure and Deploying BOSH and Ops Manager to Azure Manually.

The procedures below involve using the Ops Manager API. For more information about the Ops Manager API, see the documentation at `https://OPS-MAN-FQDN/docs` .

Step 1: Configure Authentication

Perform the procedures in the Step 1: Access Ops Manager section of the *Configuring Ops Manager Director on Azure* topic. Set up your Ops Manager authentication system, but do not continue to the Step 2: Azure Config Page section.

Continue to the Internal Authentication or External Identity Provider section below depending on which authentication system you configured.

Internal Authentication

If you configured your Ops Manager for Internal Authentication, perform the following steps:

1. SSH into the Ops Manager VM:

   ```
   $ ssh -i opsman ubuntu@OPS_MAN_FQDN
   ```

   If the private key that you generated in the Launching an Ops Manager Director Instance on Azure without an ARM Template topic is not named `opsman` , provide the correct filename instead.

2. From the Ops Manager VM, use the User Account and Authentication Command Line Interface (UAAC) to target your Ops Manager UAA server:

   ```
   $ uaac target https://YOUR-OPS-MAN-FQDN/uaa
   ```

   > **Note**: UAA is the Cloud Foundry identity and management service. See the User Account and Authentication (UAA) Server topic for more information.

3. Retrieve your token to authenticate:

   ```
   $ uaac token owner get
   Client ID: opsman
   Client secret: [Leave Blank]
   User name: OPS-MAN-USERNAME
   Password: OPS-MAN-PASSWORD
   ```

4. Continue to Step 2: Configure Ops Manager.

External Identity Provider

If you configured your Ops Manager for an external Identity Provider with SAML, perform the following steps:

1. From your local machine, target your Ops Manager UAA server:

   ```
   $ uaac target https://YOUR-OPS-MAN-FQDN/uaa
   ```

2. Retrieve your token to authenticate. When prompted for a passcode, retrieve it from `https://YOUR-OPS-MAN-FQDN/uaa/passcode` .

```
$ uaac token sso get
Client ID: opsman
Client secret: [Leave Blank]
Passcode: YOUR-PASSCODE
```

If authentication is successful, the UAAC displays the following message: `Successfully fetched token via owner password grant.`

3. Continue to Step 2: Configure Ops Manager.

Step 2: Configure Ops Manager

Perform the following steps to configure Ops Manager for Azure Germany:

1. List your tokens.

   ```
   $ uaac contexts
   ```

   Locate the entry for your Ops Manager FQDN. Under `client_id: opsman`, record the value for `access_token`.

2. Use `curl` to pass in the `environment:AzureGermanCloud` key to Ops Manager using the API:

   ```
   $ curl "https://YOUR-OPS-MAN-FQDN/api/v0/staged/director/properties" \
    -X PUT \
    -H "Authorization: Bearer UAA-ACCESS-TOKEN" \
    -H "Content-Type: application/json" \
    -d '{
     "iaas_configuration": {
       "subscription_id": "SUBSCRIPTION-ID",
       "tenant_id": "TENANT-ID",
       "client_id": "APPLICATION-ID",
       "resource_group_name": "$RESOURCE-GROUP",
       "bosh_storage_account_name": "$STORAGE-NAME",
       "deployments_storage_account_name": "MY-DEPLOYMENT-STORAGE-X",
       "default_security_group": "opsmgr-nsg",
       "ssh_public_key": "ssh-rsa OPS-MAN-PUBLIC-KEY",
       "ssh_private_key": "-----BEGIN EXAMPLE RSA PRIVATE KEY-----\OPS-MAN-PRIVATE-KEY",
       "environment": "AzureGermanCloud"
     },
     "director_configuration": {
       "ntp_servers_string": "us.pool.ntp.org",
       "metrics_ip": "1.2.3.4",
       "resurrector_enabled": true,
       "max_threads": 1,
       "database_type": "internal",
       "blobstore_type": "local"
     },
     "security_configuration": {
     "trusted_certificates": "—— BEGIN SSL CERTIFICATE —— ... ",
     "generate_vm_passwords": true
    }
   }'
   ```

   Replace the placeholder values under `iaas_configuration` as follows:
   - `UAA_ACCESS_TOKEN` is the token you retrieved from `uaac contexts` in the previous step.
   - `SUBSCRIPTION_ID`, `TENANT_ID`, and `APPLICATION_ID` are set in the Preparing to Deploy PCF on Azure topic.
   - The `$RESOURCE_GROUP` and `$STORAGE_NAME` environment variables are set in the Launching an Ops Manager Director Instance on Azure without an ARM Template topic.
   - For the values for `MY_DEPLOYMENT_STORAGE_X`, `OPS_MAN_PUBLIC_KEY`, and `OPS_MAN_PRIVATE_KEY`, see the procedures in the Launching an Ops Manager Director Instance on Azure without an ARM Template topic.

   Leave the values under `director_configuration` and `security_configuration`. They are designed to be overridden in a subsequent step when you configure Ops Manager using the web interface.

3. If the `curl` command returns a `200 OK` response, navigate to the Ops Manager FQDN in a browser and log in.

4. Configure the Ops Manager Director tile by performing the procedures in the Configuring Ops Manager Director on Azure topic.

5. When Ops Manager finishes deploying, you can optionally deploy BOSH Add-Ons to your system, and then install one or more PCF runtime environments to complete your PCF installation.

Troubleshooting PCF on Azure

Page last updated:

This topic describes how to troubleshoot known issues when deploying Pivotal Cloud Foundry (PCF) on Azure.

Troubleshoot Installation Issues

Slow Performance or Timeouts

Symptom

Developers suffer from slow performance or timeouts when pushing or managing apps, and end users suffer from slow performance or timeouts when accessing apps

Explanation

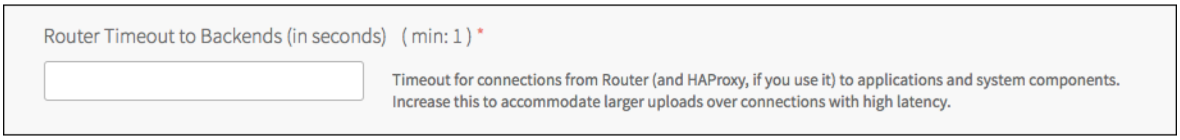
The Azure Load Balancer (ALB) disconnects active TCP connections lying idle for over four minutes.

Solution

To mitigate slow performance or timeouts, the default value of the **Router Timeout to Backends (in seconds)** field is set to 900 seconds. This default value is set high to mitigate performance issues but operators should tune this parameter to fit their infrastructure.

To edit the **Router Timeout to Backends (in seconds)** field:

1. Select the Pivotal Application Service (PAS) tile that is located within your **Installation Dashboard**.
2. Select the **Networking** tab.
3. Enter your desired time, in seconds, within the **Router Timeout to Backends (in seconds)** field.



4. Click **Save**.

Cannot Copy the Ops Manager Image

Symptom

Cannot copy the Ops Manager image into your storage account when completing Step 2: Copy Ops Manager Image of the *Launching an Ops Manager Director Instance with an ARM Template* topic or Step 4: Boot Ops Manager of the *Launching an Ops Manager Director Instance on Azure without an ARM Template* topic.

Explanation

You have an outdated version of the Azure CLI. You need the Azure CLI version 2.0.0 or greater. Run `az --version` from the command line to display your current Azure CLI version.

Solution

Install the Azure CLI 2.0 by following the instructions for your operating system in the Azure documentation.

Deployment Fails at “create-env”

Symptom

After clicking **Apply Changes** to install Ops Manager and PAS, the deployment fails at `create-env` with an error message similar to the following:

```
Command 'deploy' failed:
  Deploying:
    Creating instance 'bosh/0':
      Waiting until instance is ready:
        Starting SSH tunnel:
          Parsing private key file '/tmp/bosh_ec2_private_key.pem':
            asn1: structure error: tags don't match (16 vs {class:3 tag:28 length:127
            isCompound:false}) {optional:false explicit:false application:false
            defaultValue:<nil> tag:<nil> stringType:0 timeType:0 set:false omitEmpty:false} pkcs1PrivateKey @2
            ===== 2016-09-29 16:28:22 UTC Finished "bosh create-env"
            /var/tempest/workspaces/default/deployments/bosh.yml";
            Duration: 328s; Exit Status: 1
Exited with 1.
```

Explanation

You provided a passphrase when creating your key pair in the Step 2: Copy Ops Manager Image section of the *Launching an Ops Manager Director Instance with an ARM Template* topic or Step 4: Boot Ops Manager section of the *Launching an Ops Manager Director Instance on Azure without an ARM Template* topic.

Solution

Create a new key pair with no passphrase and redo the installation, beginning with the step for creating a VM against the Ops Manager image in the Step 2: Copy Ops Manager Image section of the *Launching an Ops Manager Director Instance with an ARM Template* topic or the Step 4: Boot Ops Manager section of the *Launching an Ops Manager Director Instance on Azure without an ARM Template* topic.

Deleting PCF from Azure

Page last updated:

When you deploy Pivotal Cloud Foundry (PCF) to Azure, you provision a set of resources. This topic describes how to delete the resources associated with a PCF deployment.

The fastest way to remove resources is to delete the resource group, or resource groups, associated with your PCF on Azure installation.

Delete the Resource Group

Perform the following steps to delete a resource group:

1. Navigate to the Azure Portal.

2. Within your subscription, select **Resource Groups**.

3. Click on the resource group you wish to delete.

4. In the details pane for the resource group, click on the trash can icon. Review the information in the confirmation screen before proceeding.

5. To confirm deletion, type in the resource group name and click **Delete**.

For more information about managing resource groups in Azure, see the Azure documentation.

Upgrading Ops Manager Director on Azure

Page last updated:

This topic describes how to upgrade Ops Manager Director for Pivotal Cloud Foundry (PCF) on Azure.

Follow the procedures below as part of the upgrade process documented in the Upgrading Pivotal Cloud Foundry topic.

Note: The Azure portal sometimes displays the names of resources with incorrect capitalization. Always use the Azure CLI to retrieve the correctly capitalized name of a resource.

Step 1: Export Environment Variables

1. Install the Azure CLI 2.0 by following the instructions for your operating system in the Azure documentation.

2. Set your cloud:

```
$ az cloud set --name AzureCloud
```

   If you deployed PCF in an environment other than Azure Cloud, consult the following list:
   - For Azure China, replace `AzureCloud` with `AzureChinaCloud`. If logging in to `AzureChinaCloud` fails with a `CERT_UNTRUSTED` error, use the latest version of node, 4.x or later.
   - For Azure Government Cloud, replace `AzureCloud` with `AzureUSGovernment`.
   - For Azure Germany, replace `AzureCloud` with `AzureGermanCloud`.

3. Log in:

```
$ az login
```

   Authenticate by navigating to the URL in the output, entering the provided code, and clicking your account.

4. Ensure that the following environment variables are set to the names of the resources you created when originally deploying Ops Manager by following the procedures in either the Launching an Ops Manager Director Instance with an ARM Template topic or the Launching an Ops Manager Director Instance on Azure without an ARM Template topic.
   - `$RESOURCE_GROUP`: This should be set to the name of your resource group. Run `az group list` to list the resource groups for your subscription.
   - `$LOCATION`: This should be set to your location, such as `westus`. For a list of available locations, run `az account list-locations`.
   - `$STORAGE_NAME`: This should be set to your BOSH storage account name. Run `az storage account list` to list your storage accounts.

5. Retrieve the connection string for the account.

```
$ az storage account show-connection-string \
--name $STORAGE_NAME --resource-group $RESOURCE_GROUP
```

   The command returns output similar to the following:

```
{
 "connectionString": "DefaultEndpointsProtocol=https;EndpointSuffix=core.windows.net;AccountName=cfdocsboshstorage;AccountKey=EXAMPLEaaaaaaaaaleiIy/Lprnc5igFxYWsgq(
}
```

6. From the `data:` field in the output above, record the full value of `connectionString` from the output above, starting with and including `DefaultEndpointsProtocol=`.

7. Export the value of `connectionString` as the environment variable `$AZURE_STORAGE_CONNECTION_STRING`.

```
$ export AZURE_STORAGE_CONNECTION_STRING="YOUR-ACCOUNT-KEY-STRING"
```

Step 2: Set Up Ops Manager

1. Navigate to Pivotal Network and download the release of **Pivotal Cloud Foundry Ops Manager for Azure** you want to upgrade to.

2. View the downloaded PDF and locate the Ops Manager image URL appropriate for your region.

3. Export the Ops Manager image URL as an environment variable.

```
$ export OPS_MAN_IMAGE_URL="YOUR-OPS-MAN-IMAGE-URL"
```

This command overrides the old Ops Manager image URL with the new Ops Manager image URL.

4. Copy the Ops Manager image into your storage account.

```
$ az storage blob copy start --source-uri $OPS_MAN_IMAGE_URL \
--connection-string $AZURE_STORAGE_CONNECTION_STRING \
--destination-container opsmanager \
--destination-blob image.vhd
```

5. Copying the image may take several minutes. Run the following command and examine the output under `"copy"`:

```
$ az storage blob show --name image.vhd \
--container-name opsmanager \
--account-name $STORAGE_NAME
...
"copy": {
  "completionTime": "2017-06-26T22:24:11+00:00",
  "id": "b9c8b272-a562-4574-baa6-f1a04afcefdf",
  "progress": "53687091712/53687091712",
  "source": "https://opsmanagerwestus.blob.core.windows.net/images/ops-manager-1.11.3.vhd",
  "status": "success",
  "statusDescription": null
},
```

When `status` reads `success`, continue to the next step.

Step 3: Configure IP Address

You have two choices for the Ops Manager IP address. Choose one of the following:

- Reuse the existing dynamic public IP address.
- Use a new dynamic public IP address.

Reuse Existing Dynamic Public IP Address

1. List your VMs and record the name of your Ops Manager VM:

```
$ az vm list
```

2. Delete your old Ops Manager VM:

```
$ az vm delete --name YOUR-OPS-MAN-VM --resource-group $RESOURCE_GROUP
```

3. List your network interfaces and record the name of the Ops Manager network interface:

```
$ az resource list --resource-group $RESOURCE_GROUP \
--resource-type Microsoft.Network/networkInterfaces
```

Use a New Dynamic Public IP Address

1. Create a new public IP address named `ops-manager-ip-new`.

```
$ az network public-ip create --name ops-manager-ip-new \
--resource-group $RESOURCE_GROUP --location $LOCATION \
--allocation-method Static
{
 "publicIp": {
  "dnsSettings": null,
  "etag": "W/\"4450ebe2-9e97-4b17-9cf2-44838339c661\"",
  "id": "/subscriptions/995b7eed-77ef-45ff-a5c9-1a405ffb8243/resourceGroups/cf-docs/providers/Microsoft.Network/publicIPAddresses/ops-manager-ip-new",
  "idleTimeoutInMinutes": 4,
  "ipAddress": "40.83.148.183",
  "ipConfiguration": null,
  "location": "westus",
  "name": "ops-manager-ip-new",
  "provisioningState": "Succeeded",
  "publicIpAddressVersion": "IPv4",
  "publicIpAllocationMethod": "Static",
  "resourceGroup": "cf-docs",
  "resourceGuid": "950d4831-1bec-42da-8a79-959bcddea9dd",
  "tags": null,
  "type": "Microsoft.Network/publicIPAddresses"
 }
}
```

2. Record the value for `ipAddress` from the output above. This is the public IP address of Ops Manager.

3. Create a network interface for Ops Manager.

```
$ az network nic create --vnet-name pcf-net \
--subnet pcf --network-security-group opsmgr-nsg \
--private-ip-address 10.0.0.5 \
--public-ip-address ops-manager-ip-new \
--resource-group $RESOURCE_GROUP \
--name ops-manager-nic-new --location $LOCATION
```

4. Shut down your old Ops Manager VM:

```
$ az vm stop --name ops-manager --resource-group $RESOURCE_GROUP
```

   If your Ops Manager VM is not named `ops-manager`, provide the correct name. To list all VMs in your account, use `az vm list`.

5. Update your DNS record to point to your new public IP address of Ops Manager.

Step 4: Boot Ops Manager

1. If you want to use the keypair from your previous Ops Manager, locate the path to the file on your local machine. If you want to create a new keypair, enter the following command:

```
$ ssh-keygen -t rsa -f opsman -C ubuntu
```

   When prompted for a passphrase, press the `enter` key to provide an empty passphrase.

2. Create a managed disk from the Ops Manager image:

```
$ az disk create --resource-group $RESOURCE_GROUP \
--name opsman-disk \
--source https://$STORAGE_NAME.blob.core.windows.net/opsmanager/image.vhd \
--location $LOCATION --size-gb 120
```

   If you are using Azure China, Azure Government Cloud, or Azure Germany, replace `blob.core.windows.net` with the following:

   - For Azure China, use `blob.core.chinacloudapi.cn`. See the Azure documentation for more information.
   - For Azure Government Cloud, use `blob.core.usgovcloudapi.net`. See the Azure documentation for more information.
   - For Azure Germany, use `blob.core.cloudapi.de`. See the Azure documentation for more information.

3. Create your Ops Manager VM, replacing `PATH-TO-PUBLIC-KEY` with the path to your public key `.pub` file.

```
$ az vm create --name ops-manager --resource-group $RESOURCE_GROUP \
--location $LOCATION --os-type linux \
--nics ops-manager-nic \
--attach-os-disk opsman-disk \
--admin-username ubuntu \
--size Standard_DS2_v2 \
--ssh-key-value PATH-TO-PUBLIC-KEY
```

PCF on GCP Requirements

Page last updated:

This guide describes how to install Pivotal Cloud Foundry (PCF) on Google Cloud Platform (GCP).

To view production-level deployment options for PCF on GCP, see the Reference Architecture for Pivotal Cloud Foundry on GCP.

General Requirements

The following are general requirements for deploying and managing a PCF deployment with Ops Manager and Pivotal Application Service (PAS):

- A wildcard DNS record that points to your router or load balancer. Alternatively, you can use a service such as xip.io. For example, `203.0.113.0.xip.io`.

  PAS gives each application its own hostname in your app domain. With a wildcard DNS record, every hostname in your domain resolves to the IP address of your router or load balancer, and you do not need to configure an A record for each app hostname. For example, if you create a DNS record `*.example.com` pointing to your load balancer or router, every application deployed to the `example.com` domain resolves to the IP address of your router.
- At least one wildcard TLS certificate that matches the DNS record you set up above, `*.example.com`.
- Sufficient IP allocation:
  - One static IP address for either HAProxy or one of your gorouters
  - One IP address for each VM instance
  - An additional IP address for each compilation worker
    So the formula for total IPs needed is `IPs needed = static IPs + VM instances + compilation workers`

  > **Note**: Pivotal recommends that you allocate at least 36 dynamic IP addresses when deploying Ops Manager and PAS. BOSH requires additional dynamic IP addresses during installation to compile and deploy VMs, install PAS, and connect to services.
- One or more NTP servers if not already provided by your IaaS.
- (**Recommended**) A network without DHCP available for deploying the PAS VMs.

  > **Note**: If you have DHCP, refer to the Troubleshooting Guide to avoid issues with your installation.
- (**Optional**) External storage. When you deploy PCF, you can select internal file storage or external file storage, either network-accessible or IaaS-provided, as an option in the PAS tile. Pivotal recommends using external storage whenever possible. See Upgrade Considerations for Selecting File Storage in Pivotal Cloud Foundry for a discussion of how file storage location affects platform performance and stability during upgrades.
- (**Optional**) External databases. When you deploy PCF, you can select internal or external databases for the BOSH Director and for PAS. Pivotal recommends using external databases in production deployments.
- (**Optional**) External user stores. When you deploy PCF, you can select a SAML user store for Ops Manager or a SAML or LDAP user store for PAS, to integrate existing user accounts.
- The most recent version of the Cloud Foundry Command Line Interface (cf CLI).

GCP Requirements

You must have the following to install PCF on GCP:

- A GCP project with sufficient quota to deploy all the VMs needed for a PCF installation. For a list of suggested quotas, see Recommended GCP Quotas.

  You can request a quota increase on the GCP Quotas page.
- A GCP account with adequate permissions to create resources within the selected GCP project. Per the Least Privileged User principle, the permissions required to set up a GCP environment for PCF include:
  - Permissions to create firewalls, networks, load balancers, and other resources:
    - **Compute Engine** > **Compute Instances Admin (beta)**
    - **Compute Engine** > **Compute Network Admin**
    - **Compute Engine** > **Compute Security Admin**

- If using Google Cloud Storage (GCS) for Cloud Controller file storage, permissions to create buckets:
  - **Storage** > **Storage Admin**
- If you are using Cloud DNS, permissions to add and modify DNS entries:
  - **Project** > **Editor**

Note: When you deploy PCF, the deployment processes run under a separate service account with the minimum permissions required to install Ops Manager and Pivotal Application Service (PAS).

- The Google Cloud SDK installed on your machine and authenticated to your GCP account.
- Sufficiently high instance limits, or no instance limits, on your GCP account. The exact number of instances depends on the number of tiles and availability zones you plan to deploy. At a minimum, a new GCP deployment requires the following `custom` VMs:

VM Count	Machine type	Memory (in GB)
30	1vCPU	1.00
3	1vCPU	2.00
4	2vCPU	4.00
3	2vCPU	8.00
3	4vCPU	16.00

By default, PAS deploys the number of VM instances required to run a highly available configuration of PCF. If you are deploying a test or sandbox PCF that does not require HA, then you can scale down the number of instances in your deployment. For information about the number of instances required to run a minimal, non-HA PCF deployment, see Scaling PAS.

- Administrative rights to a domain for your PCF installation. You need to be able to add wildcard records to this domain. You specify this registered domain when configuring the SSL certificate and Cloud Controller for your deployment. For more information see the Providing a Certificate for your SSL Termination Point topic.
- An SSL certificate for your PCF domain. This can be a self-signed certificate, which Ops Manager can generate for you, but Pivotal recommends using a self-signed certificate for development and testing purposes only. If you plan to deploy PCF into a production environment, you must obtain a certificate from your Certificate Authority.

Certificate Requirements on GCP

If you are deploying PCF on GCP, then you must add your certificate to both the frontend configuration of your HTTP Load Balancer and to the Gorouter (PAS Router). For more information, see Create Instance Groups and the HTTP(S) Load Balancer.

GCP load balancers actually forward both encrypted (WebSockets) and unencrypted (HTTP and TLS-terminated HTTPS) traffic to the Gorouter. When configuring the point-of-entry for a GCP deployment, select **Forward SSL to PAS Router** in your PAS network configuration. This point-of-entry selection accommodates this special characteristic of GCP deployments.

See Certificate Requirements for general certificate requirements for deploying PCF.

GCP Permissions Guidelines

Pivotal recommends following the principle of least privilege by scoping privileges to the most restrictive permissions possible for a given role. SeeIaaS Permissions Guidelines for recommendations on how to create and scope GCP accounts for PCF.

GCP Security Documents

- GCP authentication documentation
  This developer-facing documentation explains general authentication guidelines for GCP.

Install PCF on GCP

Install PCF on GCP Manually

Complete the following procedures to install PCF on GCP:

1. Preparing to Deploy PCF on GCP
2. Launching an Ops Manager Director Instance on GCP
3. Configuring Ops Manager Director on GCP
4. (Optional) Configuring a Shared VPC on GCP
5. (Optional) Installing the PCF IPsec Add-On
6. Deploying PAS on GCP

Install PCF on GCP Using Terraform

Complete the following procedures to install PCF on GCP:

1. Preparing to Deploy PCF on GCP (Terraform)
2. Configuring Ops Manager Director on GCP (Terraform)
3. (Optional) Configuring a Shared VPC on GCP
4. (Optional) Installing the PCF IPsec Add-On
5. Deploying Pivotal Application Service (PAS) on GCP (Terraform)

Delete PCF on GCP

You can use the GCP console to remove an installation of all components, but retain the objects in your bucket for a future deployment:

- Deleting a GCP Installation from the Console

Troubleshoot PCF on GCP

The troubleshooting document for PCF on GCP infrastructure.

- Troubleshooting PCF on GCP

Recommended GCP Quotas

Page last updated:

Default quotas on a new GCP subscription do not have enough quota for a typical production-level PCF deployment.

The following table lists recommended resource quotas for a single PCF deployment.

Metric	Resource	Suggested Minimum Quota
CPUs	Regional	150*
Firewall rules	Global	15
Forwarding rules	Global	15
Backend services	Global	5
Health checks	Global	5
Images	Global	10
Static IP addresses**	Regional	5
In-use IP addresses	Global	5
In-use IP addresses**	Regional	5
Networks	Global	5
Subnetworks	Global	5
Routes	Global	20
Target pools	Global	5
Target HTTP proxies	Global	5
Target HTTPS proxies	Global	5
Persistent Disk Standard (GB)	Regional	15,000

* Assuming a deployment with 100 app instances.
** Assuming a SNAT topology.

To view production-level deployment options for PCF on GCP, see the Reference Architecture for Pivotal Cloud Foundry on GCP.

For instructions on how to set up GCP resources required to deploy PCF, see Preparing to Deploy PCF on GCP.

Installing PCF on GCP Manually

Complete the following procedures to install PCF on GCP manually:

1. Preparing to Deploy PCF on GCP
2. Launching an Ops Manager Director Instance on GCP
3. Configuring Ops Manager Director on GCP
4. (Optional) Configuring a Shared VPC on GCP
5. (Optional) Installing the PCF IPsec Add-On
6. Deploying Pivotal Application Service (PAS) on GCP

Pivotal

Preparing GCP

Page last updated:

This guide describes the preparation steps required to install Pivotal Cloud Foundry (PCF) on Google Cloud Platform (GCP).

In addition to fulfilling the prerequisites listed in the Installing Pivotal Cloud Foundry on GCP topic, you must create resources in GCP such as a new network, firewall rules, load balancers, and a service account before deploying PCF. Follow these procedures to prepare your GCP environment.

When deploying PCF on GCP, Pivotal recommends using the following GCP components:

- Google Cloud SQL for external Pivotal Application Service (PAS) database services
- NAT Gateway Instances to limit the number of VMs with public IP addresses
- Google Cloud Storage for external filestorage in PAS

For more information, review the different deployment options and recommendations in the Reference Architecture for Pivotal Cloud Foundry on GCP.

Step 1: Set up an IAM Service Account

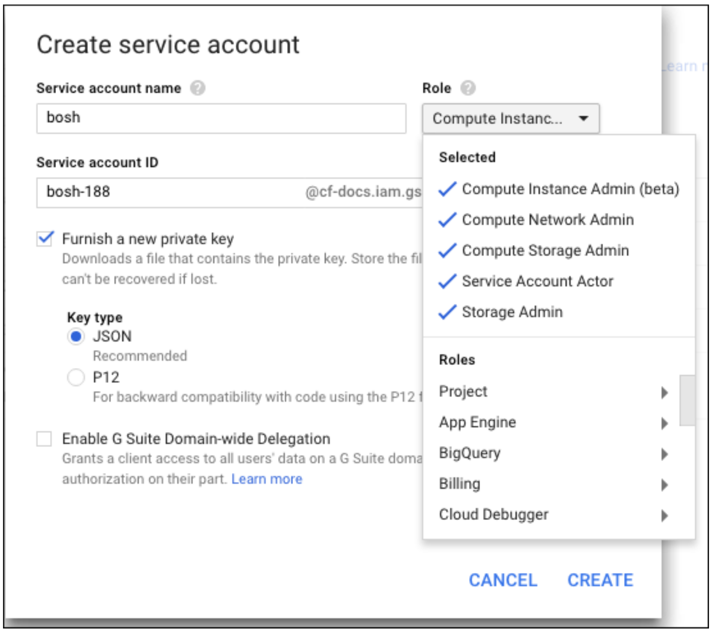
1. From the GCP Console, select **IAM & Admin**, then **Service accounts**.
2. Click **Create Service Account**:
   - **Service account name**: Enter a name. For example, `bosh`.
   - **Role**: Select the following roles for the service account:

     > You must scroll down in the pop-up windows to select all required roles.

     - **Project** > **Service Account Actor**
     - **Compute Engine** > **Compute Instance Admin**
     - **Compute Engine** > **Compute Network Admin**
     - **Compute Engine** > **Compute Storage Admin**
     - **Storage** > **Storage Admin**

     > The **Service Account Actor** role is only required if you plan to use **The Ops Manager VM Service Account** to deploy Ops Manager.

   - **Service account ID**: The field automatically generates a unique ID based on the username.
   - **Furnish a new private key**: Select this checkbox and JSON as the **Key type**.



3. Click **Create**. Your browser automatically downloads a JSON file with a private key for this account. Save this file in a secure location.

Step 2: Enable Google Cloud APIs

Ops Manager manages GCP resources using the Google Compute Engine and Cloud Resource Manager APIs. To enable these APIs, perform the following steps:

1. Log in to the Google Developers console at https://console.developers.google.com.
2. In the console, navigate to the GCP project where you want to install PCF.
3. Select **API Manager > Library**.
4. Under **Google Cloud APIs**, select **Compute Engine API**.
5. On the **Google Compute Engine API** page, click **Enable**.
6. In the search field, enter `Google Cloud Resource Manager API`.
7. On the **Google Cloud Resource Manager API** page, click **Enable**.
8. To verify that the APIs have been enabled, perform the following steps:
    a. Log in to GCP using the IAM service account you created in Set up an IAM Service Account:

    ```
    $ gcloud auth activate-service-account --key-file JSON_KEY_FILENAME
    ```

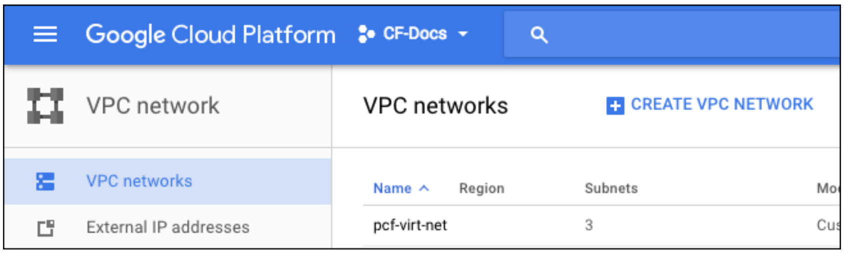
    b. List your projects:

    ```
    $ gcloud projects list
    PROJECT_ID      NAME            PROJECT_NUMBER
    my-project-id   my-project-name  ##############
    ```

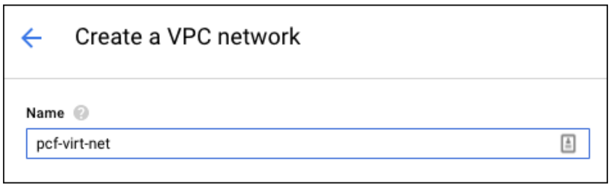
    This command lists the projects where you enabled Google Cloud APIs.

Step 3: Create a GCP Network with Subnets

1. Log in to the GCP Console.
2. Navigate to the GCP project where you want to install PCF.
3. Select **VPC network**, then **CREATE VPC NETWORK**.



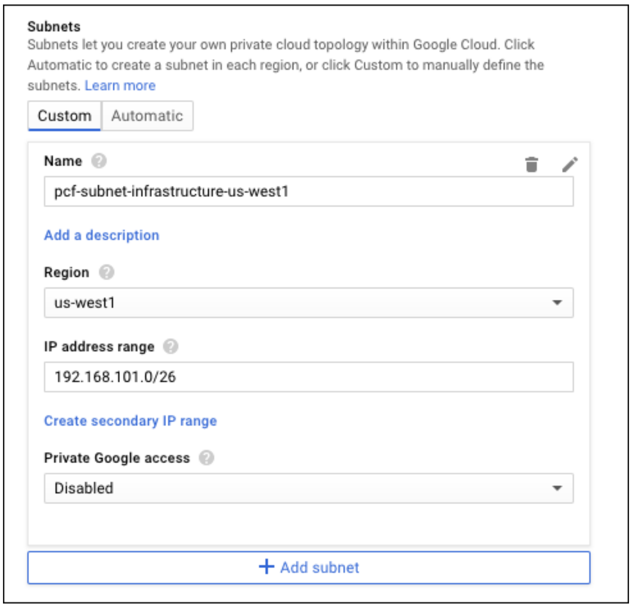
4. In the **Name** field, enter `MY-PCF-virt-net`.
   `MY-PCF` is a prefix to help you identify resources for this PCF deployment in the GCP console.



    a. Under **Subnets** complete the form as follows:

Name	`MY-PCF-subnet-infrastructure-MY-GCP-REGION`
Region	A region that supports three availability zones. For help selecting the correct region for your deployment, see the Google documentation on regions and zones.

IP address range	A CIDR ending in `/26`



See the following image for an example:

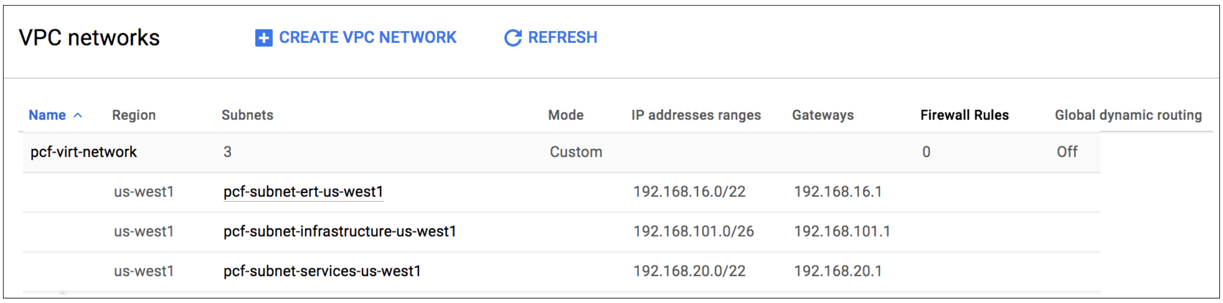
b. Click **Add subnet** to add a second subnet with the following details:

Name	`MY-PCF-subnet-ert-MY-GCP-REGION`
Region	The same region you selected for the infrastructure subnet
IP address range	A CIDR ending in `/22` Example: `192.168.16.0/22`

c. Click **Add subnet** to add a third **Subnet** with the following details:

Name	`MY-PCF-subnet-services-MY-GCP-REGION`
Region	The same region you selected for the previous subnets
IP address range	A CIDR in `/22` Example: `192.168.20.0/22`

See the following image for an example:



5. Under **Dynamic routing mode**, leave **Regional** selected.

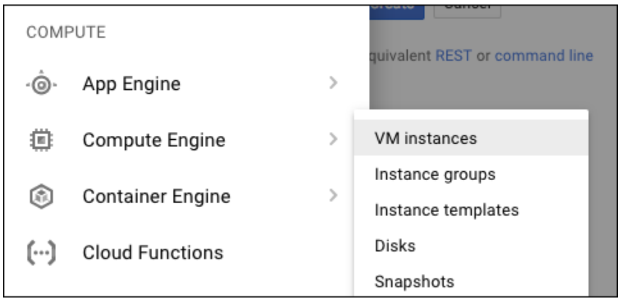
6. Click **Create**.

Step 4: Create NAT Instances

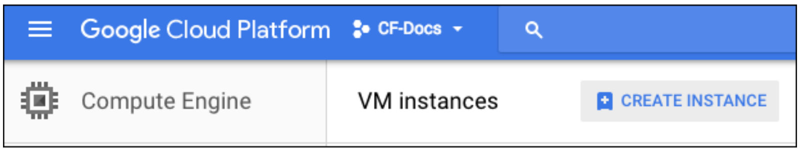
Use NAT instances when you want to expose only a minimal number of public IP addresses.

For more information, see the Reference Architecture for Pivotal Cloud Foundry on GCP and GCP documentation.

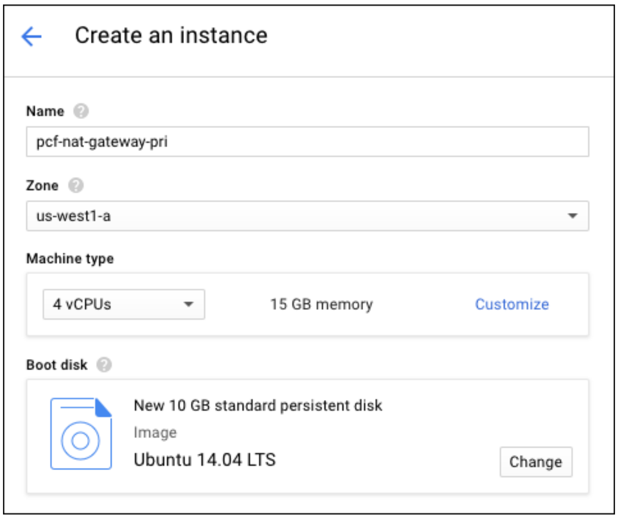
1. In the console, navigate to **Compute Engine** > **VM instances**.



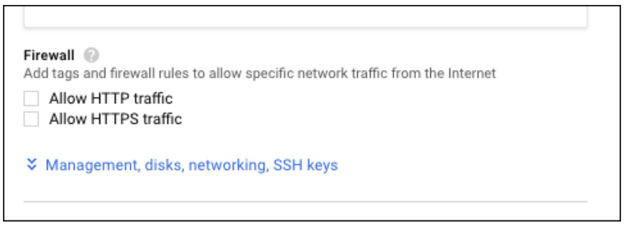
2. Click **CREATE INSTANCE**.



3. Complete the following fields:
   - **Name**: Enter `MY-PCF-nat-gateway-pri`.
     This is the first, or primary, of three NAT instances you need.
   - **Zone**: Select the first zone from your region.
     Example: For region `us-west1`, select zone `us-west1-a`.
   - **Machine type**: Select `n1-standard-4`.
   - **Boot disk**: Click **Change** and select `Ubuntu 14.04 LTS`.

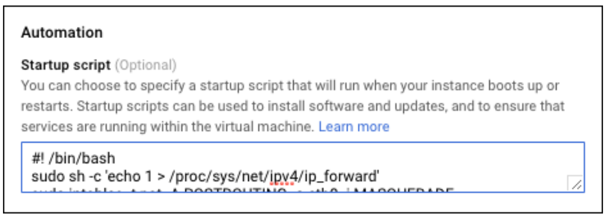


4. Expand the additional configuration fields by clicking **Management, disks, networking, SSH keys**.

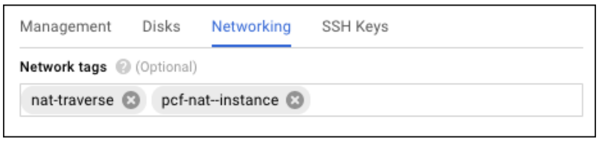


   a. In the **Startup script** field under **Automation**, enter the following text:

```
#! /bin/bash
sudo sh -c 'echo 1 > /proc/sys/net/ipv4/ip_forward'
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```



5. Click **Networking** to open additional network configuration fields:



   a. In the **Network tags** field, add the following: `nat-traverse` and `MY-PCF-nat-instance`.
   b. Click the pencil icon to edit the **Network interface**.
   c. For **Network**, select `MY-PCF-virt-net`. You created this network in Step 1: Create a GCP Network with Subnets.
   d. For **Subnetwork**, select `MY-PCF-subnet-infrastructure-MY-GCP-REGION`.
   e. For **Primary internal IP**, select `Ephemeral (Custom)`. Enter an IP address (for example, `192.168.101.2`) in the **Custom ephemeral IP address** field. Specify an internal IP address located within the reserved IP range that you will configure in Ops Manager Director. Do not use the **Gateway** IP, for example `192.168.101.1`.
   f. For **External IP**, select `Ephemeral`.

   > **Note**: If you select a static external IP address for the NAT instance, then you can use the static IP to further secure access to your CloudSQL instances.

   g. Set **IP forwarding** to `On`.
   h. Click **Done**.

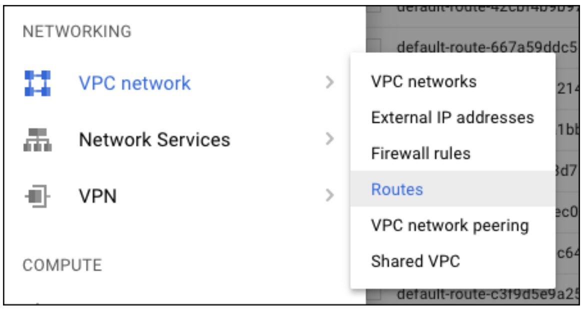
6. Click **Create** to finish creating the NAT instance.

7. Repeat steps 2-6 to create two additional NAT instances with the names and zones specified in the table below. The rest of the configuration remains the same.

Instance 2	**Name**: `MY-PCF-nat-gateway-sec`. **Zone**: Select the second zone from your region. Example: For region `us-west1`, select zone `us-west1-b`. **Internal IP**: Select `Custom` and enter an IP address (for example, `192.168.101.3`) in the **Internal IP address** field. Specify an internal IP address located within the reserved IP range that you will configure in Ops Manager Director. Do not use the **Gateway** IP, for example `192.168.101.1`.
Instance 3	**Name**: `MY-PCF-nat-gateway-ter`. **Zone**: Select the third zone from your region. Example: For region `us-west1`, select zone `us-west1-c`. **Internal IP**: Select `Custom` and enter an IP address (for example, `192.168.101.4`) in the **Internal IP address** field. Specify an internal IP address located within the reserved IP range that you will configure in Ops Manager Director. Do not use the **Gateway** IP, for example `192.168.101.1`.

Create Routes for NAT Instances

1. In the GCP console, navigate to **VPC Networks** > **Routes**.



2. Click **CREATE ROUTE**.

3. Complete the form as follows:
   - **Name**: `MY-PCF-nat-pri`
   - **Network**: `MY-PCF-virt-net`
   - **Destination IP range**: `0.0.0.0/0`
   - **Priority**: `800`
   - **Instance tags**: `MY-PCF`
   - **Next hop**: `Specify an instance`
   - **Next hop instance**: `MY-PCF-nat-gateway-pri`

4. Click **Create** to finish creating the route.

5. Repeat steps 2-4 to create two additional routes with the names and next hop instances specified in the table below. The rest of the configuration remains the same.

Route 2	Name: `MY-PCF-nat-sec` Next hop instance: `MY-PCF-nat-gateway-sec`
Route 3	Name: `MY-PCF-nat-ter` Next hop instance: `MY-PCF-nat-gateway-ter`

Step 5: Create Firewall Rules for the Network

GCP lets you assign tags to virtual machine (VM) instances and create firewall rules that apply to VMs based on their tags. This step assigns tags and firewall rules to Ops Manager components and VMs that handle incoming traffic.

1. In the **Networking** pane, select **Firewall rules**.

2. Create firewall rules according to the table below:

> **Note:** If you want your firewalls rules to only allow traffic within your private network, modify the **Source IP Ranges** from the table accordingly.

Firewall Rules	
Rule 1	This rule allows SSH from public networks. **Name:** `MY-PCF-allow-ssh` **Network:** `MY-PCF-virt-net` **Allowed protocols and ports:** `tcp:22` **Source filter:** IP ranges **Source IP ranges:** `0.0.0.0/0` **Target tags:** `allow-ssh`
Rule 2	This rule allows HTTP from public networks. **Name:** `MY-PCF-allow-http` **Network:** `MY-PCF-virt-net` **Allowed protocols and ports:** `tcp:80` **Source filter:** IP ranges **Source IP ranges:** `0.0.0.0/0` **Target tags:** `allow-http`, `router`
Rule 3	This rule allows HTTPS from public networks. **Name:** `MY-PCF-allow-https` **Network:** `MY-PCF-virt-net` **Allowed protocols and ports:** `tcp:443` **Source filter:** IP ranges **Source IP ranges:** `0.0.0.0/0` **Target tags:** `allow-https`, `router`
Rule 4	This rule allows Gorouter health checks. **Name:** `MY-PCF-allow-http-8080` **Network:** `MY-PCF-virt-net` **Allowed protocols and ports:** `tcp:8080` **Source filter:** IP ranges **Source IP Ranges:** `0.0.0.0/0` **Target tags:** `router`
Rule 5	This rule allows communication between BOSH-deployed PAS jobs. **Name:** `MY-PCF-allow-ert-all` **Network:** `MY-PCF-virt-net`

	Allowed protocols and ports: `tcp;udp;icmp` **Source filter:** Source tags **Target tags:** `MY-PCF`, `MY-PCF-opsman`, `nat-traverse` **Source tags:** `MY-PCF`, `MY-PCF-opsman`, `nat-traverse`
Rule 6 (Optional)	This rule allows access to the TCP router. **Name:** `MY-PCF-allow-cf-tcp` **Network:** `MY-PCF-virt-net` **Source filter:** IP ranges **Source IP ranges:** `0.0.0.0/0` **Allowed protocols and ports:** `tcp:1024-65535` **Target tags:** `MY-PCF-cf-tcp`
Rule 7 (Optional)	This rule allows access to the SSH proxy. **Name:** `MY-PCF-allow-ssh-proxy` **Network:** `MY-PCF-virt-net` **Source filter:** IP ranges **Source IP ranges:** `0.0.0.0/0` **Allowed protocols and ports:** `tcp:2222` **Target tags:** `MY-PCF-ssh-proxy`, `diego-brain`

3. If you are only using your GCP project to deploy PCF, then you can delete the following default firewall rules:
   - `default-allow-http`
   - `default-allow-https`
   - `default-allow-icmp`
   - `default-allow-internal`
   - `default-allow-rdp`
   - `default-allow-ssh`

Step 6: Create Database Instance and Databases

Create Database Instance

1. From the GCP Console, select **SQL** and click **CREATE INSTANCE**.

2. Ensure **MySQL** is selected and click **Next**.

3. Click **Choose Second Generation**.

4. Configure the instance as follows:
   - **Instance ID**: `MY-PCF-ert-sql`
   - **Root password**: Set a password for the root user.
   - **Region**: Select the region you specified when creating networks.
   - **Zone**: `Any`.
   - **Configure machine type and storage**: Select `db-n1-standard-2`.
   - **Enable auto backups and high availability**: Make the following selections:
     - Leave **Automate backups** and **Enable binary logging** selected.
     - Under **High availability**, select the **Create failover replica** checkbox.
   - **Authorize Networks**: Click **Add network** and create a network named `all` that allows traffic from `0.0.0.0/0`.

   > **Note:** If you assigned static IP addresses to your NAT instances, you can instead limit access to the database instances by specifying the NAT IP addresses.

5. Click **Create**.

Create Databases

1. From the **Instances** page, select the database instance you just created.

2. Select the **Databases** tab.

3. Use the **Create database** button to create the following databases:
   - `account`
   - `app_usage_service`
   - `autoscale`
   - `ccdb`
   - `console`
   - `diego`
   - `locket`
   - `networkpolicyserver`
   - `nfsvolume`
   - `notifications`
   - `routing`
   - `silk`
   - `uaa`
   - `credhub`

4. Select the **USERS** tab.

5. Use the **Create user account** button to create a unique username and password for each database you created above. For **Host name**, select **Allow any host**. You must create a total of 13 user accounts.

Step 7: Create Storage Buckets

1. From the GCP Console, select **Storage** > **Browser**.

2. Using **CREATE BUCKET**, create buckets with the following names. For **Default storage class**, select **Multi-Regional**:
   - `MY-PCF-buildpacks`
   - `MY-PCF-droplets`
   - `MY-PCF-packages`
   - `MY-PCF-resources`

Step 8: Create HTTP Load Balancer

Create Instance Group

1. Navigate to **Compute Engine** > **Instance groups**.

2. Click **CREATE INSTANCE GROUP**.

3. Complete the form as follows:
   - For **Name**, enter `MY-PCF-http-lb`
   - For **Location**, select **Single-zone**.
   - For **Zone**, select the first zone from your region.
     Example: For region `us-west1`, select zone `us-west1-a`.
   - Under **Group type**, select **Unmanaged instance group**.
   - For **Network**, select `MY-PCF-virt-net`.
   - For **Subnetwork**, select the `MY-PCF-subnet-ert-MY-GCP-REGION` subnet that you created previously.
   - Click **Create**.

4. Create a second instance group with the following details:
   - **Name**: `MY-PCF-http-lb`
   - **Location**: **Single-zone**
   - **Zone**: Select the second zone from your region.
     Example: For region `us-west1`, select zone `us-west1-b`.
   - **Group type**: Select **Unmanaged instance group**.
   - **Network**: Select `MY-PCF-virt-net`.
   - **Subnetwork**: Select the `MY-PCF-subnet-ert-MY-GCP-REGION` subnet that you created previously.

5. Create a third instance group with the following details:
   - **Name**: `MY-PCF-http-lb`
   - **Location**: **Single-zone**
   - **Zone**: Select the third zone from your region.
     Example: For region `us-west1`, select zone `us-west1-c`.
   - **Group type**: Select **Unmanaged instance group**.
   - **Network**: Select `MY-PCF-virt-net`.
   - **Subnetwork**: Select the `MY-PCF-subnet-ert-MY-GCP-REGION` subnet that you created previously.

Create Health Check

1. Navigate to **Compute Engine** > **Health checks**.

2. Click **CREATE HEALTH CHECK**.

3. Complete the form as follows:
   - **Name**: `MY-PCF-cf-public`
   - **Port**: `8080`
   - **Request path**: `/health`
   - **Check interval**: `30`
   - **Timeout**: `5`
   - **Healthy threshold**: `10`
   - **Unhealthy threshold**: `2`

4. Click **Create**.

Configure Backend

1. Navigate to **Network services** > **Load balancing**.

2. Click **CREATE LOAD BALANCER**.

3. Under **HTTP(S) Load Balancing**, click the **Start configuration** button.

4. For the **Name**, enter `MY-PCF-global-pcf`.

5. Select **Backend configuration**

   a. From the dropdown, select **Backend services** > **Create a backend service**.
   b. Complete the form as follows:
   c. **Name**: `MY-PCF-http-lb-backend`.
   d. **Protocol**: `HTTP`.
   e. **Named port**: `http`.
   f. **Timeout**: `10 seconds`.
   g. Under **Backends** > **New backend**, select the **Instance group** that corresponds to the first zone of the multi-zone instance group you created. For example: `MY-PCF-http-lb-us-west1-a`. Click **Done**.
   h. Click **Add backend**, select the **Instance group** that corresponds to the second zone of the multi-zone instance group you created. For example: `MY-PCF-http-lb-us-west1-b`. Click **Done**.
   i. Click **Add backend**, select the **Instance group** that corresponds to the third zone of the multi-zone instance group you created. For example: `MY-PCF-http-lb-us-west1-c`. Click **Done**.
   j. **Health check**: Select the `MY-PCF-cf-public` health check that you created.
   k. **Cloud CDN**: Ensure Cloud CDN is disabled.
   l. Click **Create**.

Configure Frontend

1. Click **Host and path rules** to populate the default fields and a green check mark.

2. Select **Frontend configuration**, and add the following:
   - **Name**: `MY-PCF-cf-lb-http`
   - **Protocol**: `HTTP`
   - **IP**: Perform the following steps:

    1. Select **Create IP address**.
    2. Enter a **Name** for the new static IP address and an optional description. For example, `MY-PCF-global-pcf`.
    3. Click **Reserve**.

  - **Port:** `80`

3. If you are using a trusted SSL certificate or already have a self-signed certificate, proceed to step 5.

4. If you want to use a self-signed certificate generated during PAS network configuration, skip over the next step of adding the HTTPS frontend configuration until after you generate the certificate in PAS. After you generate the certificate, return to step 5 using the following guidelines:
   - Copy and paste the generated contents of the **Router SSL Termination Certificate and Private Key** fields from PAS into the public certificate and private key fields.
   - Since you are using a self-signed certificate, do not enter a value in the **Certificate Chain** field.

5. Click **Add Frontend IP and port** and add the following:
   - **Name:** `MY-PCF-cf-lb-https`
   - **Protocol:** `HTTPS`
   - **IP address:** Select the `MY-PCF-global-pcf` address you create for the previous **Frontend IP and Port**.
   - **Port:** `443`
   - **Certificate:** Select **Create a new certificate**. In the next dialog, perform the following steps:

     - In the **Name** field, enter a name for the certificate.
     - In the **Public key certificate** field, copy in the contents of your public certificate, or upload your certificate as a .pem file.
     - In the **Certificate chain** field, enter or upload your certificate chain in the .pem format. If you are using a self-signed certificate, you do not need to populate this field.
     - In the **Private key** field, copy in the contents or upload the .pem file of the private key for the certificate.

6. Review the completed frontend configuration.

7. Click **Review and finalize** to verify your configuration.

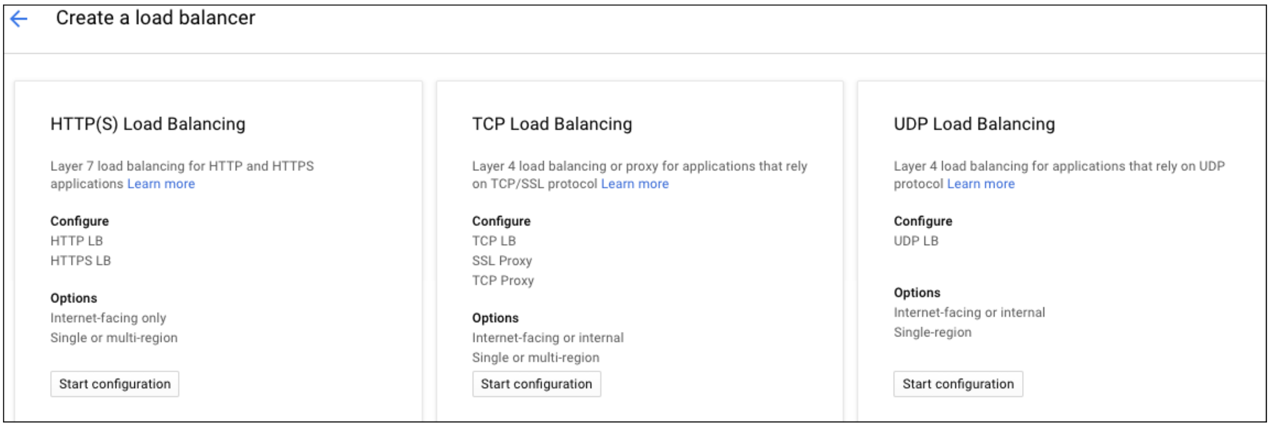
8. Click **Create**.

Step 9: Create TCP WebSockets Load Balancer

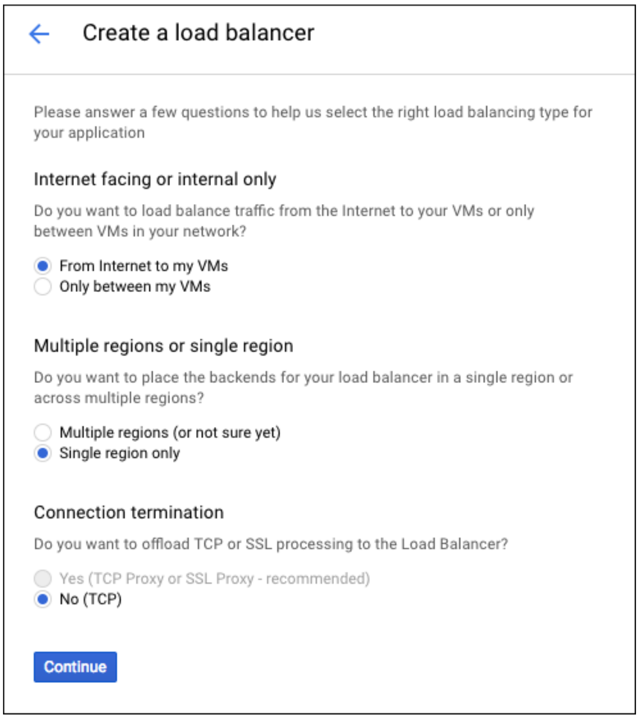
The load balancer for tailing logs with WebSockets for PCF on GCP operates on TCP port `443`.

1. From the GCP Console, select **Network services > Load balancing > Create load balancer**.

2. Under **TCP Load Balancing**, click **Start configuration**.



3. In the **Create a load balancer** configuration screen, make the following selections:
   - Under **Internet facing or internal only**, select **From Internet to my VMs**.
   - Under **Multiple regions or single region**, select **Single region only**.

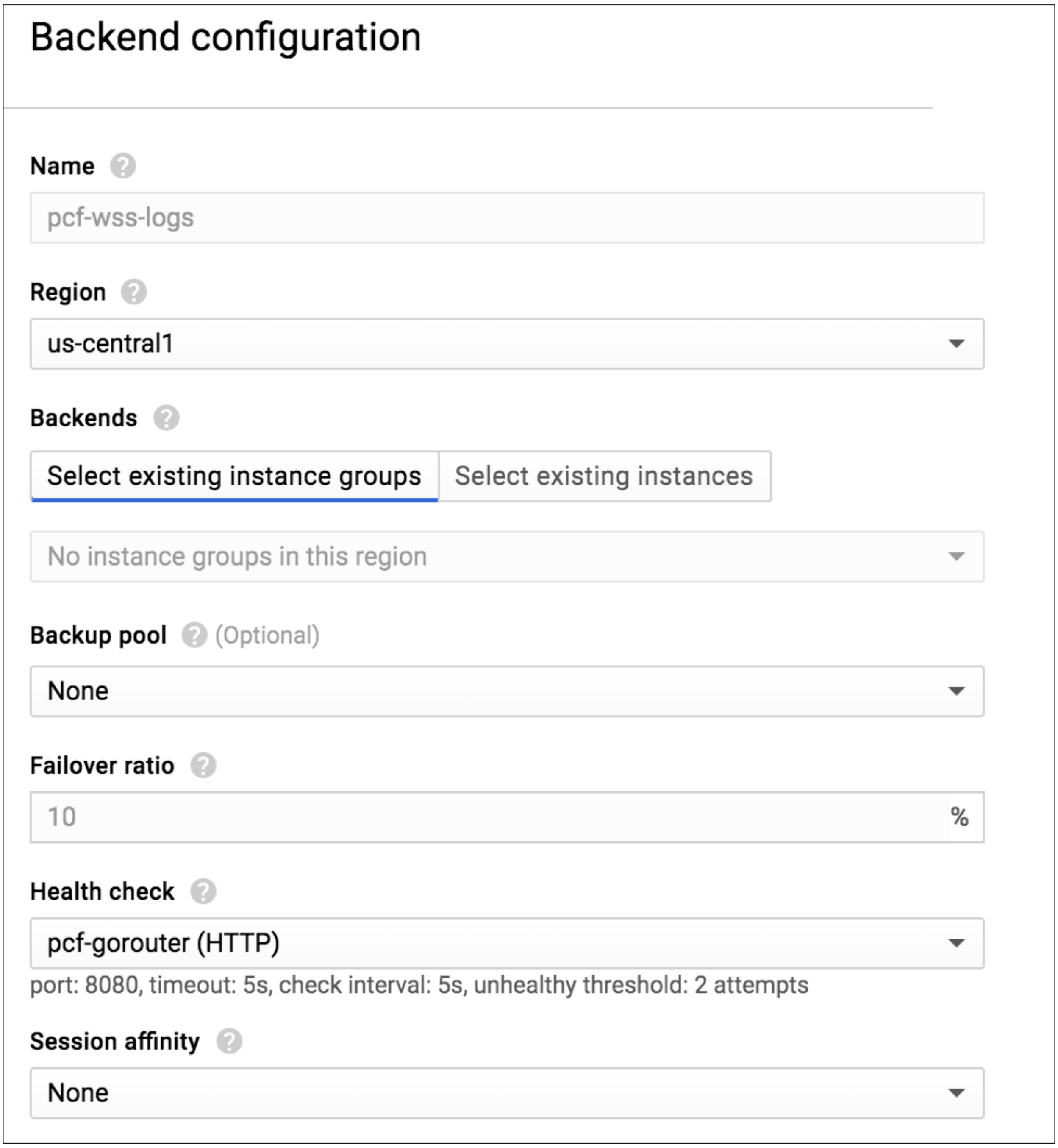


   - Under **Connection termination**, select **No (TCP)**.

4. Click **Continue**.

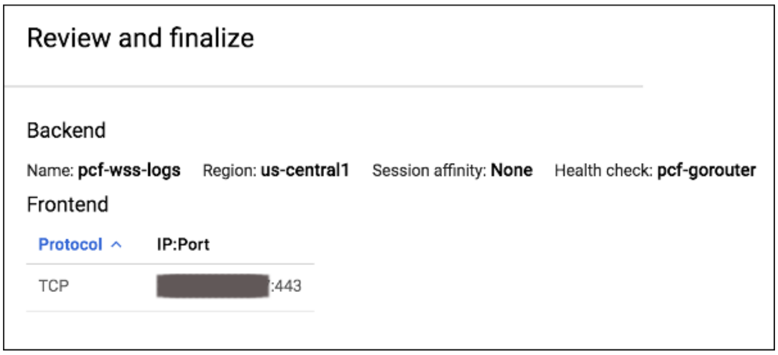
5. In the **New TCP load balancer** window, enter `MY-PCF-wss-logs` in the **Name** field.

6. Click **Backend configuration** to configure the **Backend service**:



- **Region**: Select the region you used to create the network in [Create a GCP Network with Subnets](#).
- From the **Health check** drop-down menu, create a health check with the following details:
  - **Name**: `MY-PCF-gorouter`
  - **Port**: `8080`
  - **Request path**: `/health`
  - **Check interval**: `30`
  - **Timeout**: `5`
  - **Healthy threshold**: `10`
  - **Unhealthy threshold**: `2` The **Backend configuration** section shows a green check mark.

7. Click **Frontend configuration** to open its configuration window and complete the fields:
   - **Protocol**: `TCP`
   - **IP**: Perform the following steps:
     1. Select **Create IP address**.
     2. For name **Name** for the new static IP address and an optional description. For example, `MY-PCF-gorouter-wss`.
     3. Click **Reserve**.
   - **Port**: `443`



8. Click **Review and finalize** to verify your configuration.

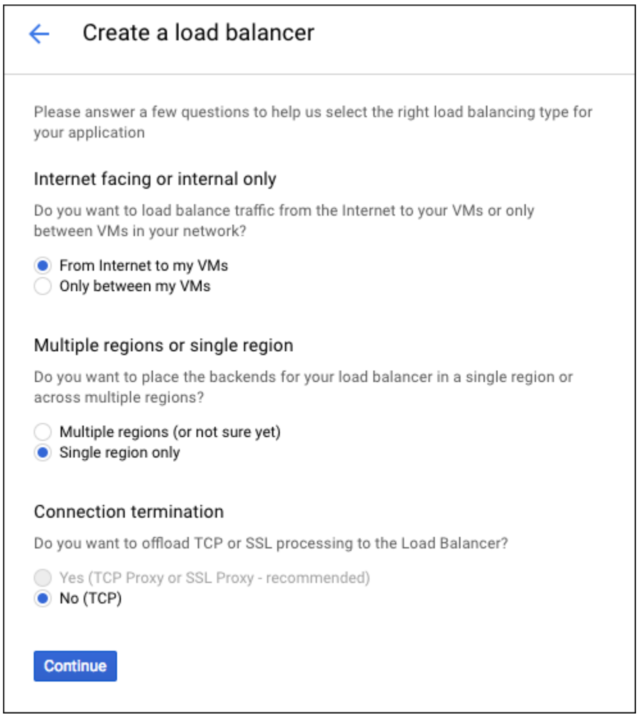
9. Click **Create**.

Step 10: Create SSH Proxy Load Balancer

1. From the GCP Console, select **Network services > Load balancing > Create load balancer**.

2. Under **TCP Load Balancing**, click **Start configuration**.

3. Under **Internet facing or internal only**, select **From Internet to my VMs**.

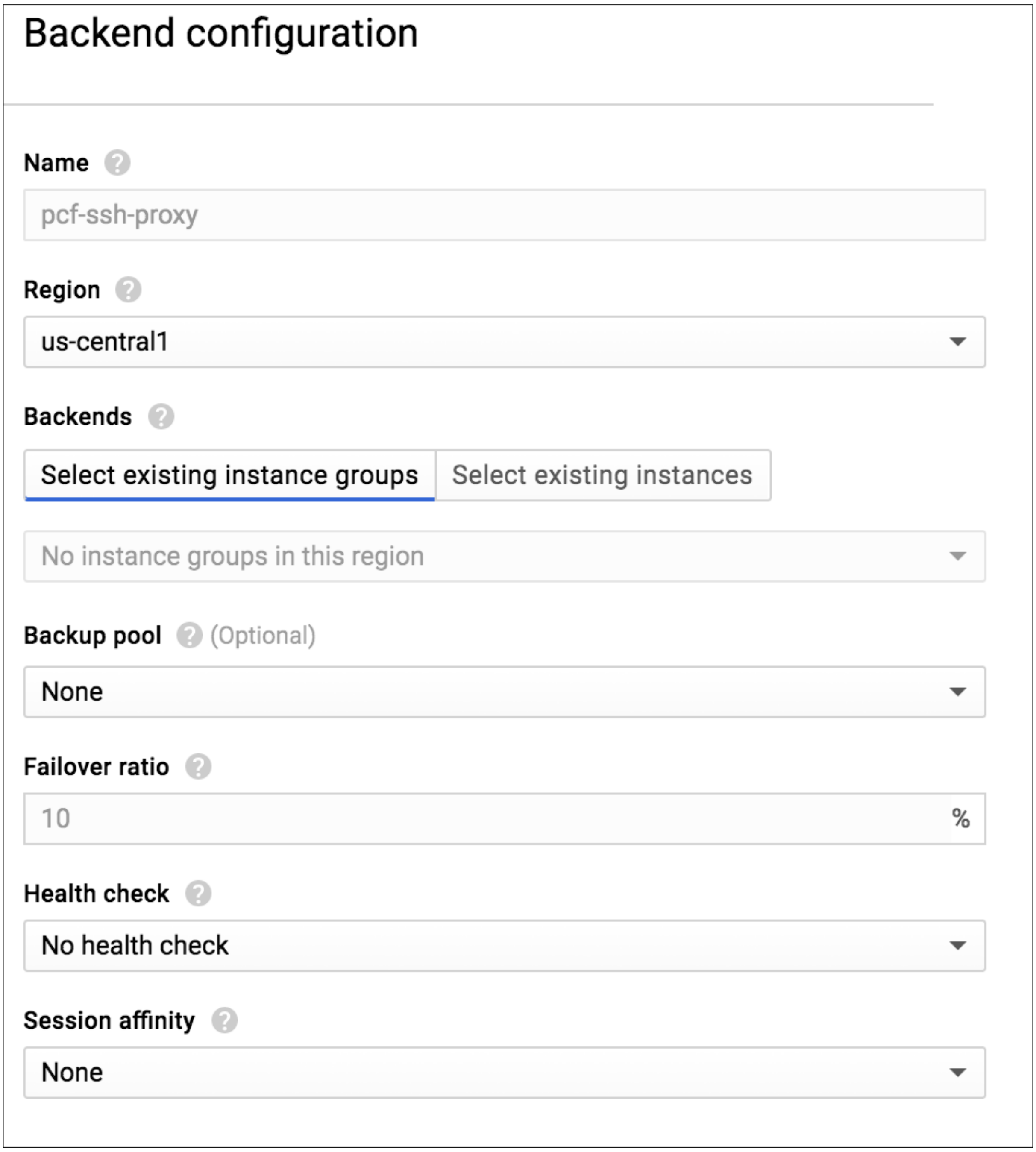


4. Under **Connection termination**, select **No (TCP)**.

5. Click **Continue**.

6. In the **New TCP load balancer** window, enter `MY-PCF-ssh-proxy` in the **Name** field.

7. Select **Backend configuration**, and enter the following field values:
   - **Region:** Select the region you used to create the network in Create a GCP Network with Subnet.
   - **Backup pool:** `None`
   - **Failover ratio:** `10%`
   - **Health check:** `No health check`



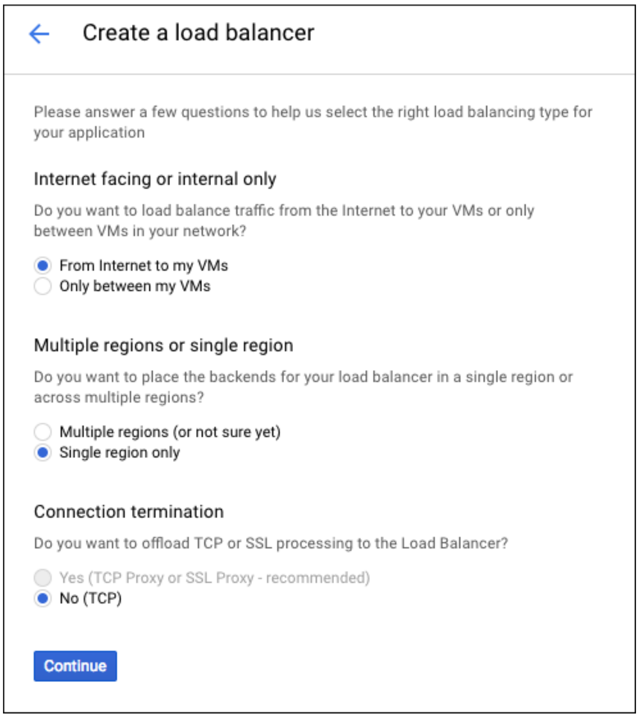
8. Select **Frontend configuration**, and add the following:
    - **Protocol:** `TCP`
    - **IP**: Perform the following steps:
        1. Select **Create IP address**.
        2. Enter a **Name** for the new static IP address and an optional description. For example, `MY-PCF-ssh-proxy`.
        3. Click **Reserve**.
    - **Port:** `2222`
9. Optionally, review and finalize your load balancer.
10. Click **Create**.

Step 11: Create Load Balancer for TCP Router

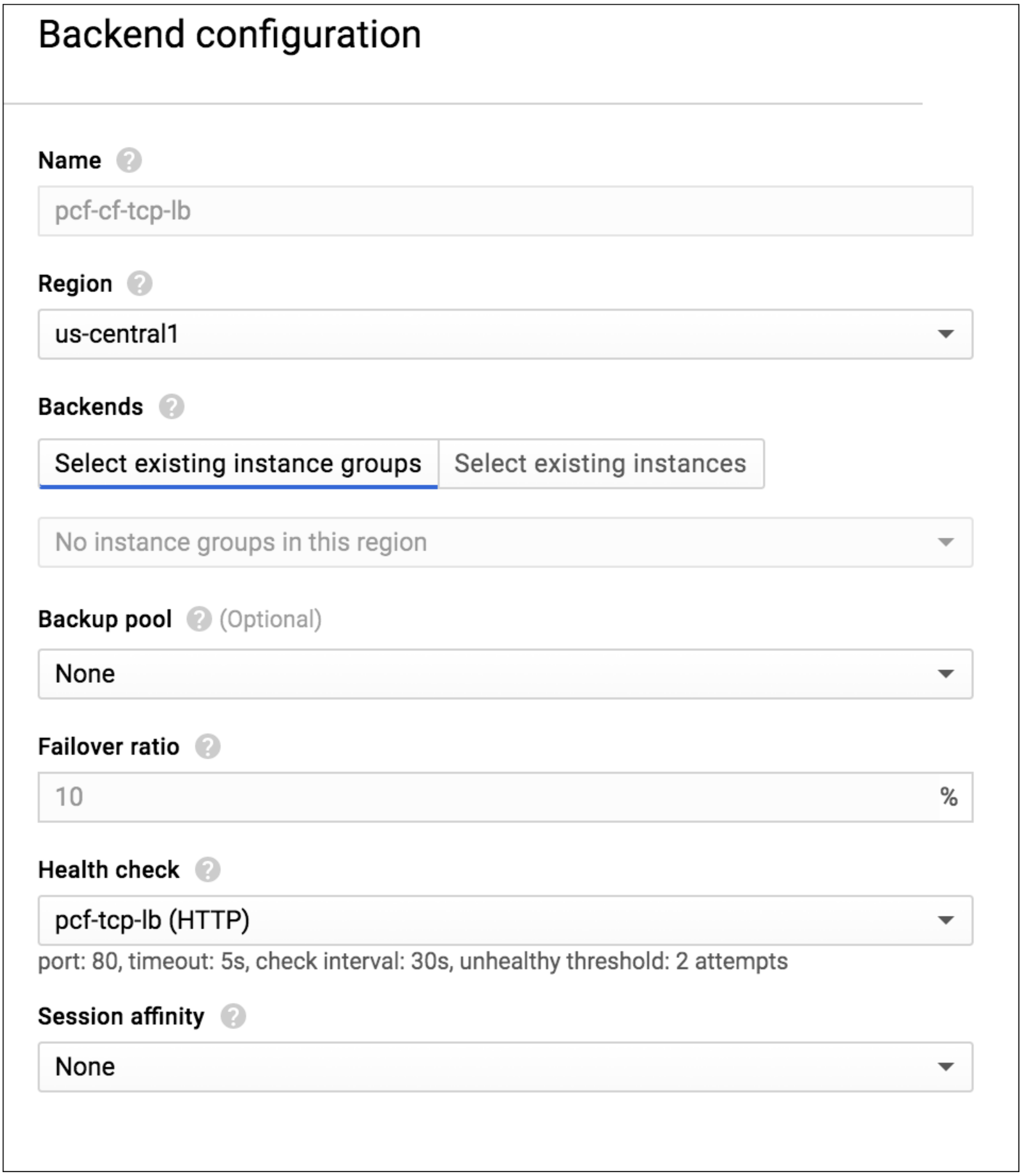
Note: This step is optional and only required if you enable TCP routing in your deployment.

To create a load balancer for TCP routing in GCP, perform the following steps:

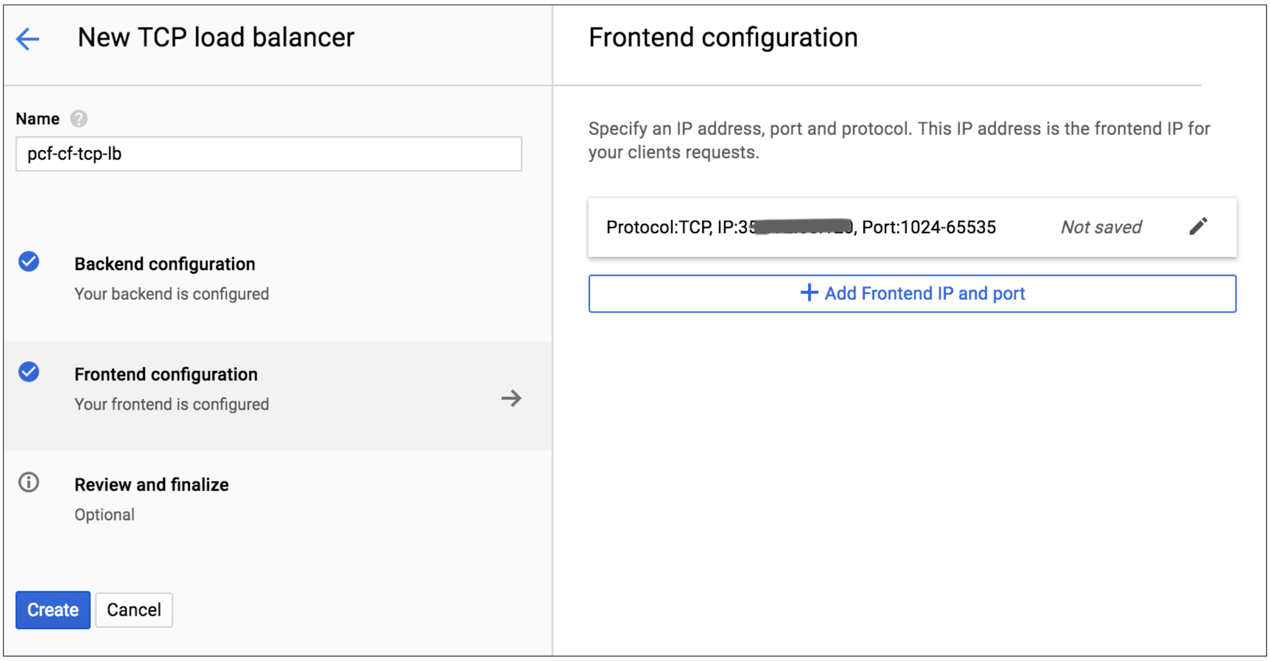
1. From the GCP Console, select **Network services > Load balancing > Create load balancer**.
2. Under **TCP Load Balancing**, click **Start configuration**.
3. Under **Connection termination**, select **No (TCP)**. Click **Continue**.



4. On the **New TCP load balancer** screen, enter a unique name for the load balancer in the **Name** field. For example, `MY-PCF-cf-tcp-lb`.
5. Select **Backend configuration**, and enter the following field values:
   - **Region**: Select the region you used to create the network in [Create a GCP Network with Subnet](#).
   - From the **Health check** drop-down menu, create a health check with the following details:
     - **Name**: `MY-PCF-tcp-lb`
     - **Port**: `80`
     - **Request path**: `/health`
     - **Check interval**: `30`
     - **Timeout**: `5`
     - **Healthy threshold**: `10`
     - **Unhealthy threshold**: `2`
     - Click **Save and continue**.



6. Select **Frontend configuration**, and add the frontend IP and port entry as follows:
    - **Protocol**: `TCP`
    - **IP**: Perform the following steps:
        1. Select **Create IP address**.
        2. Enter a **Name** for the new static IP address and an optional description. For example, `MY-PCF-cf-tcp-lb`.
        3. Click **Reserve**.
    - **Port**: `1024-65535`



7. Click **Review and finalize** to verify your configuration.

8. Click **Create**.

Step 12: Add DNS Records for Your Load Balancers

In this step you redirect queries for your domain to the IP addresses of your load balancers.

1. Locate the static IP addresses of the load balancers you created in Preparing to Deploy PCF on GCP:
   - An HTTP(S) load balancer named `MY-PCF-global-pcf`
   - A TCP load balancer for WebSockets named `MY-PCF-wss-logs`
   - A TCP load balancer named `MY-PCF-ssh-proxy`
   - A TCP load balancer named `MY-PCF-cf-tcp-lb`

   > **Note**: You can locate the static IP address of each load balancer by clicking its name under **Network services > Load balancing** in the GCP Console.

2. Log in to the DNS registrar that hosts your domain. Examples of DNS registrars include Network Solutions, GoDaddy, and Register.com.

3. Create **A records** with your DNS registrar that map domain names to the public static IP addresses of the load balancers located above:

Create and map this record...	To the IP of this load balancer	Required
`*.sys.MY-DOMAIN` Example: `*.sys.example.com`	`MY-PCF-global-pcf`	Yes
`*.apps.MY-DOMAIN` Example: `*.apps.example.com`	`MY-PCF-global-pcf`	Yes
`doppler.sys.MY-DOMAIN` Example: `doppler.sys.example.com`	`MY-PCF-wss-logs`	Yes
`loggregator.sys.MY-DOMAIN` Example: `loggregator.sys.example.com`	`MY-PCF-wss-logs`	Yes
`ssh.sys.MY-DOMAIN` Example: `ssh.sys.example.com`	`MY-PCF-ssh-proxy`	Yes, to allow SSH access to apps
`tcp.MY-DOMAIN` Example: `tcp.example.com`	`MY-PCF-cf-tcp-lb`	No, only set up if you have enabled the TCP routing feature

4. Save changes within the web interface of your DNS registrar.

5. In a terminal window, run the following `dig` command to confirm that you created your A record successfully:

```
dig xyz.EXAMPLE.COM
```

You should see the A record that you just created:

```
;; ANSWER SECTION:
xyz.EXAMPLE.COM.     1767    IN  A 203.0.113.1
```

What to Do Next

- (Optional) To save time during the final stage of the installation process, you can start downloading the PAS tile. See Step 1: Download the PAS Tile of the *Deploying PAS on GCP* topic.
- Proceed to the next step in the deployment, Launching an Ops Manager Director Instance on GCP.

Deploying BOSH and Ops Manager to GCP

Page last updated:

This topic describes how to deploy Ops Manager Director for Pivotal Cloud Foundry (PCF) on Google Cloud Platform (GCP).

After you complete this procedure, follow the instructions in the Configuring Ops Manager Director on GCP and Configuring PAS on GCP topics.

Step 1: Locate the Pivotal Ops Manager Installation File

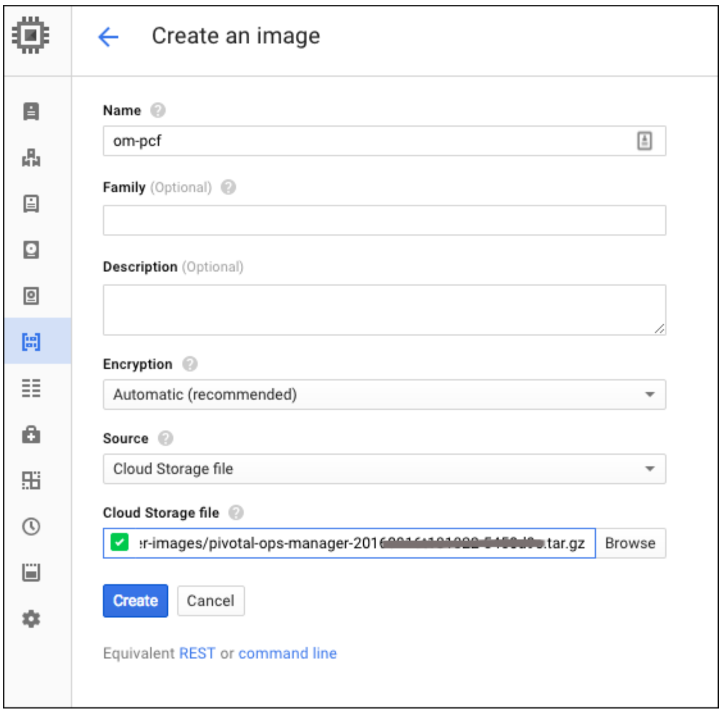
1. Log in to the Pivotal Network, and click on **Pivotal Cloud Foundry Operations Manager**.
2. From the **Releases** drop-down, select the release to install.
3. Select one of the following download files:
   - **Pivotal Cloud Foundry Ops Manager for GCP**
   - **Pivotal Cloud Foundry Ops Manager YAML for GCP**

   When you click on the download link, your browser downloads or opens the `OpsManager_version_onGCP.pdf` or `OpsManager_version_onGCP.yml` file.

   These documents provide the GCP location of the Ops Manager `.tar.gz` installation file based on the geographic location of your installation.
4. Copy the filepath string of the Ops Manager image based on your deployment location.

Step 2: Create a Private VM Image

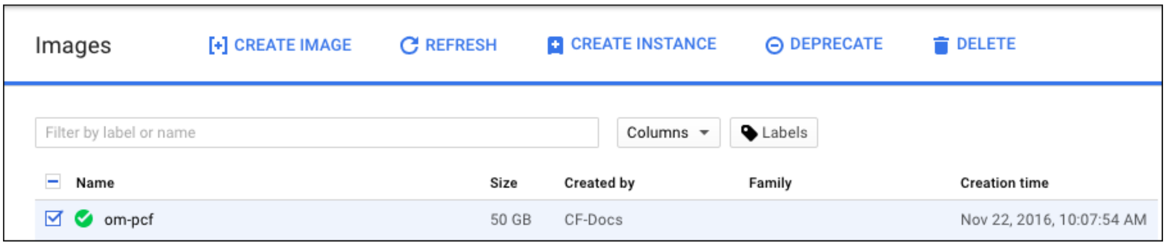
1. Log in to the GCP Console.
2. In the left navigation panel, click **Compute Engine**, and select **Images**.
3. Click **Create Image**.
4. Complete the following fields:
   - **Name**: Enter a name. For example, `opsman-pcf-gcp-2-0`.
   - **Encryption**: Leave **Automatic (recommended)** selected.
   - **Source**: Choose **Cloud Storage file**.
   - **Cloud Storage file:** Paste in the Google Cloud Storage filepath you copied from the PDF file in the previous step.



5. Click **Create**. The file may take a few minutes to import.

Step 3: Create the Ops Manager VM Instance

1. Select the checkbox for the image that you created above.



2. Click **Create Instance**.

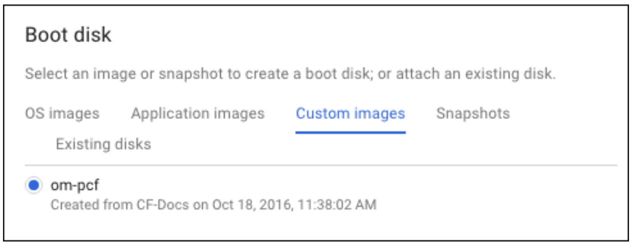
3. In the **Create an instance form**, complete the following fields:
   - **Name**: Enter a name that matches the naming conventions of your deployment.
   - **Zone**: Choose a zone from the region in which you created your network.
   - **Machine type**: Choose `n1-standard-2`.
   - Click **Customize** to manually configure the vCPU and memory. An Ops Manager VM instance requires the following minimum specifications:

Machine Spec	Minimum Value
CPU	2 vCPUs
Memory	8 GB

   - **Boot disk**: Click **Change**, then perform the following steps:
     - Click **Custom images** if it is not already selected.
     - Select the **Boot disk type**. If you have an Ops Manager environment with high performance needs, select **SSD**. As an example, environments used to develop PCF tiles may benefit from a higher performing Ops Manager VM boot disk. For most environments, however, you can select **Standard**.
     - Set the **Size (GB)** of the boot disk to the minimum or higher.

Machine Spec	Minimum Value
Boot disk	100 GB

     - Select the Ops Manager image you created in the previous step if it is not already selected.



     - Click **Select** to save.
   - Under **Identity and API access**, choose the **Service account** you created when preparing your environment during the step Set up an IAM Service Account.
   - **Allow HTTP traffic**: Leave this checkbox unselected.
   - **Allow HTTPS traffic**: Leave this checkbox unselected.
   - **Networking**: Select the **Networking** tab, and perform the following steps:
     - Under **Network interfaces**, perform the following steps:
     - Remove the `default` network interface if this interface still exists.
     - Select the network (for example, `MY-PCF-virt-network`) you created when preparing your environment in the Create a GCP Network with Subnet section of the *Preparing to Deploy PCF on GCP* topic.
     - Under **Subnetwork**, select the `MY-PCF-subnet-infrastructure-MY-GCP-REGION` subnet that you created when preparing your environment in the Create a GCP Network with Subnet section of the *Preparing to Deploy PCF on GCP* topic.
     - For **Primary internal IP**, select **Ephemeral (Custom)**. Enter an IP address (for example, `192.168.101.5`) in the **Custom ephemeral IP address** field. Specify the next available internal IP address located within the reserved IP range that you will configure in Ops Manager Director. Do not use the **Gateway IP**, for example `192.168.101.1`.
     - For **External IP**, select **Create IP address**. In the next form, enter a name for the static IP. For example, `om-public-ip`. Click **Reserve**. In

the **External IP** drop-down, select the static IP address you just reserved.
- For **Network tags**, enter `MY-PCF-opsman` and `allow-https`. These tags tag apply the firewall rules you created in Create Firewall Rules for the Network to the Ops Manager VM.

4. Click **Create** to deploy the new Ops Manager VM. This may take a few moments.

5. Navigate to your DNS provider, and create an entry that points a fully qualified domain name (FQDN) `opsman.MY-DOMAIN` to the `MY-PCF-opsman` static IP address of Ops Manager that you created in a previous step.

Note: In order to set up Ops Manager authentication correctly, Pivotal recommends using a Fully Qualified Domain Name (FQDN) to access Ops Manager. Using an ephemeral IP address to access Ops Manager can cause authentication errors upon subsequent access.

What to Do Next

After you complete this procedure, follow the instructions in the Configuring Ops Manager Director on GCP topic.

Later on, if you need to SSH into the Ops Manager VM to perform diagnostic troubleshooting, see SSH into Ops Manager.

Configuring Ops Manager Director on GCP

Page last updated:

This topic describes how to configure the Ops Manager Director for Pivotal Cloud Foundry (PCF) on Google Cloud Platform (GCP).

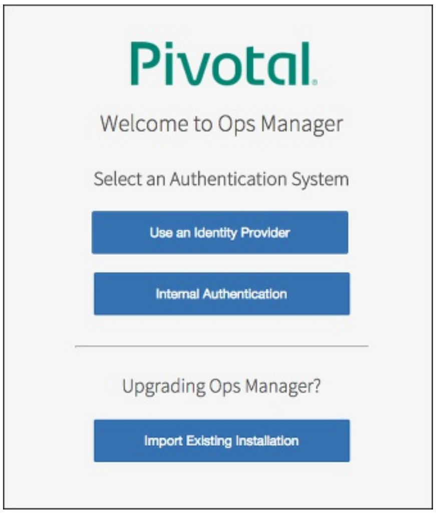
Note: You can also perform the procedures in this topic using the Ops Manager API. For more information, see the Using the Ops Manager API topic.

Step 1: Access Ops Manager

1. In a web browser, navigate to the fully qualified domain name (FQDN) of Ops Manager that you set up in Launching an Ops Manager Director Instance on GCP.

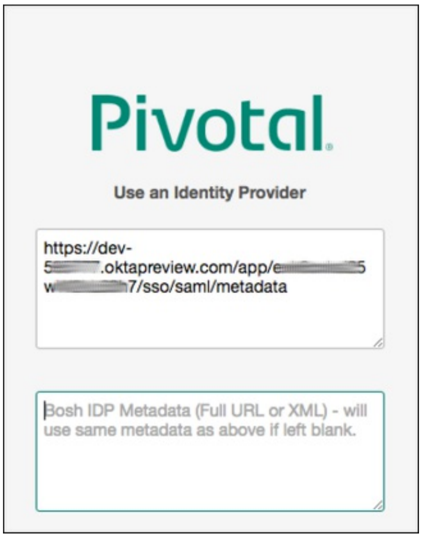
   Note: In order to set up Ops Manager authentication correctly, Pivotal recommends using a Fully Qualified Domain Name (FQDN) to access Ops Manager. Using an ephemeral IP address to access Ops Manager can cause authentication errors upon subsequent access.

2. When Ops Manager starts for the first time, you must choose one of the following:
   - Use an Identity Provider: If you use an Identity Provider, an external identity server maintains your user database.
   - Internal Authentication: If you use Internal Authentication, PCF maintains your user database.



Use an Identity Provider (IdP)

1. Log in to your IdP console and download the IdP metadata XML. Optionally, if your IdP supports metadata URL, you can copy the metadata URL instead of the XML.



2. Copy the IdP metadata XML or URL to the Ops Manager **Use an Identity Provider** log in page.

> **Note**: The same IdP metadata URL or XML is applied for the BOSH Director. If you use a separate IdP for BOSH, copy the metadata XML or URL from that IdP and enter it into the BOSH IdP Metadata text box in the Ops Manager log in page.

3. Enter your **Decryption passphrase**. Read the **End User License Agreement**, and select the checkbox to accept the terms.

4. Your Ops Manager log in page appears. Enter your username and password. Click **Login**.

5. Download your SAML Service Provider metadata (SAML Relying Party metadata) by navigating to the following URLs:
   - **5a.** Ops Manager SAML service provider metadata: `https://OPS-MAN-FQDN:443/uaa/saml/metadata`
   - **5b.** BOSH Director SAML service provider metadata: `https://BOSH-IP-ADDRESS:8443/saml/metadata`

> **Note**: To retrieve your `BOSH-IP-ADDRESS`, navigate to the **Ops Manager Director** tile > **Status** tab. Record the **Ops Manager Director** IP address.

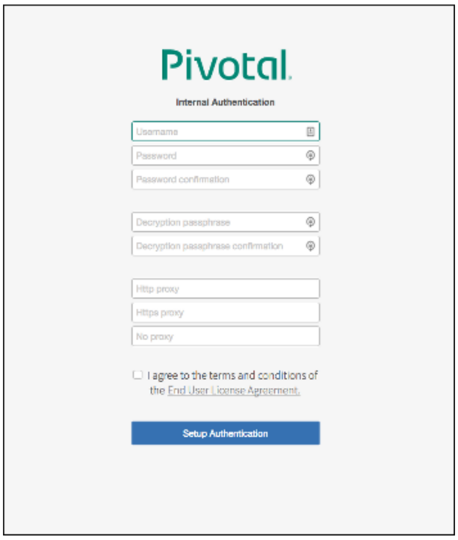
6. Configure your IdP with your SAML Service Provider metadata. Import the Ops Manager SAML provider metadata from Step 5a above to your IdP. If your IdP does not support importing, provide the values below.
   - **Single sign on** URL: `https://OPS-MAN-FQDN:443/uaa/saml/SSO/alias/OPS-MAN-FQDN`
   - **Audience URI** (SP Entity ID): `https://OP-MAN-FQDN:443/uaa`
   - **Name ID**: Email Address
   - SAML authentication requests are always signed

7. Import the BOSH Director SAML provider metadata from Step 5b to your IdP. If the IdP does not support an import, provide the values below.
   - **Single sign on** URL: `https://BOSH-IP:8443/saml/SSO/alias/BOSH-IP`
   - **Audience URI** (SP Entity ID): `https://BOSH-IP:8443`
   - **Name ID**: Email Address
   - SAML authentication requests are always signed

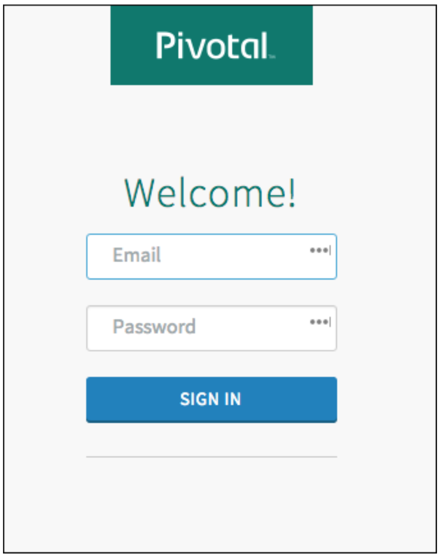
8. Return to the **Ops Manager Director** tile, and continue with the configuration steps below.

Internal Authentication

1. When redirected to the **Internal Authentication** page, you must complete the following steps:
   - Enter a **Username**, **Password**, and **Password confirmation** to create an Admin user.
   - Enter a **Decryption passphrase** and the **Decryption passphrase confirmation**. This passphrase encrypts the Ops Manager datastore, and is not recoverable if lost.
   - If you are using an **HTTP proxy** or **HTTPS proxy**, follow the instructions in the Configuring Proxy Settings for the BOSH CPI topic.
   - Read the **End User License Agreement**, and select the checkbox to accept the terms.
   - Click **Setup Authentication**.



2. Log in to Ops Manager with the Admin username and password that you created in the previous step.

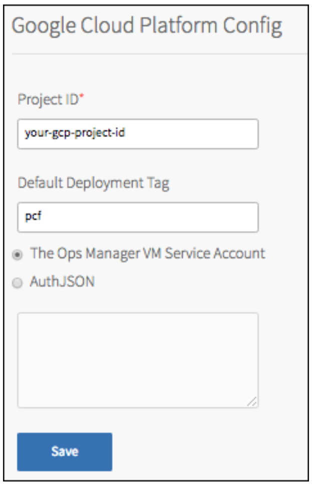


Step 2: Google Cloud Platform Config

1. Click the **Google Cloud Platform** tile within the **Installation Dashboard**.

2. Select **Google Config**. Complete the following fields:
   - **Project ID**: Enter your GCP project ID in all lower case, such as: `your-gcp-project-id`.
   - **Default Deployment Tag**: Enter the `MY-PCF` prefix that you used when creating the GCP resources for this PCF installation.
   - Select **AuthJSON** and in the field below enter the contents of the JSON file that you downloaded in the Set up an IAM Service Account section of the *Preparing to Deploy PCF on GCP* topic.

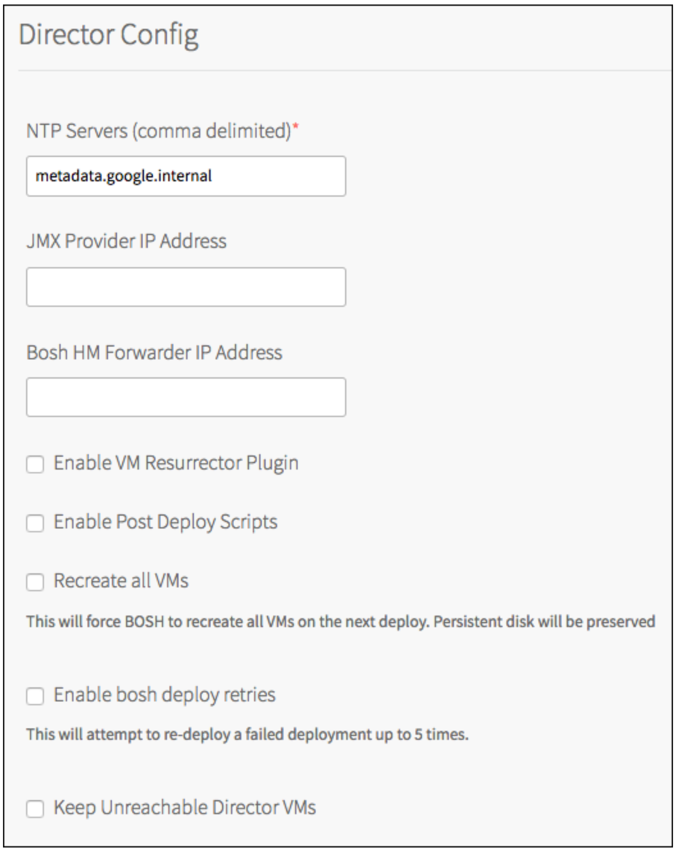
   > **Note**: As an alternative, you can select **The Ops Manager VM Service Account** option to use the service account automatically created by GCP for the Ops Manager VM. To use this option, the project-wide service account that you set up in Set up an IAM Service Account must be assigned the **Service Account Actor** role.



3. Click **Save**.

Step 3: Director Config Page

1. Select **Director Config** to open the **Director Config** page.



2. In the **NTP Servers (comma delimited)** field, enter `metadata.google.internal`.

> **Note:** To resolve `metadata.google.internal` as the NTP server hostname, you must provide the two IP addresses for DNS configuration as described in Step 5: Create Networks Page of this procedure.

3. Leave the **JMX Provider IP Address** field blank.

> **Note:** Starting from PCF v2.0, BOSH-reported system metrics are available in the Loggregator Firehose by default. Therefore, if you continue to use PCF JMX Bridge for consuming them outside of the Firehose, you may receive duplicate data. To prevent this, leave the **JMX Provider**

IP Address field blank. For additional guidance, see BOSH System Metrics Available in Loggregator Firehose.

4. Leave the **Bosh HM Forwarder IP Address** field blank.

> **Note:** Starting from PCF v2.0, BOSH-reported system metrics are available in the Loggregator Firehose by default. Therefore, if you continue to use the BOSH HM Forwarder for consuming them, you may receive duplicate data. To prevent this, leave the **Bosh HM Forwarder IP Address** field blank. For additional guidance, see BOSH System Metrics Available in Loggregator Firehose.

5. Select the **Enable VM Resurrector Plugin** checkbox to enable the Ops Manager Resurrector functionality and increase Pivotal Application Service (PAS) availability.

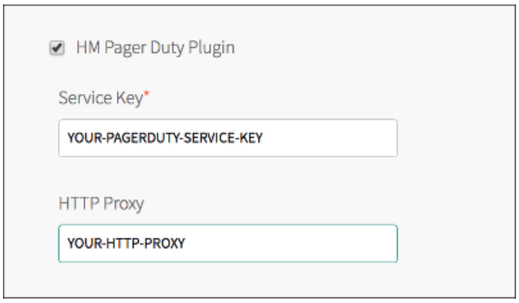
6. (Optional) Select **Enable Post Deploy Scripts** to run a post-deploy script after deployment. This script allows the job to execute additional commands against a deployment.

7. (Optional) Select **Recreate all VMs** to force BOSH to recreate all VMs on the next deploy. This process does not destroy any persistent disk data.

8. Select **Enable bosh deploy retries** for Ops Manager to retry failed BOSH operations up to five times.

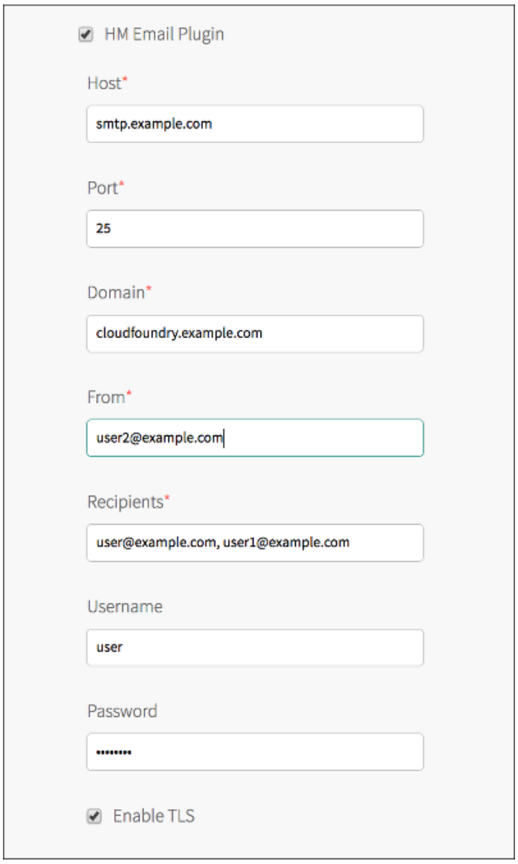
9. (Optional) Select **Keep Unreachable Director VMs** if you want to preserve Ops Manager Director VMs after a failed deployment for troubleshooting purposes.

10. (Optional) Select **HM Pager Duty Plugin** to enable Health Monitor integration with PagerDuty.



   - **Service Key**: Enter your API service key from PagerDuty.
   - **HTTP Proxy**: Enter an HTTP proxy for use with PagerDuty.

11. (Optional) Select **HM Email Plugin** to enable Health Monitor integration with email.



- **Host**: Enter your email hostname.
- **Port**: Enter your email port number.
- **Domain**: Enter your domain.
- **From**: Enter the address for the sender.
- **Recipients**: Enter comma-separated addresses of intended recipients.
- **Username**: Enter the username for your email server.
- **Password**: Enter the password password for your email server.
- **Enable TLS**: Select this checkbox to enable Transport Layer Security.

12. For **Blobstore Location**, select **Internal**.

13. For **Database Location**, select **Internal**.

14. (Optional) Modify the **Director Workers** value, which sets the number of workers available to execute Director tasks. This field defaults to `5`.

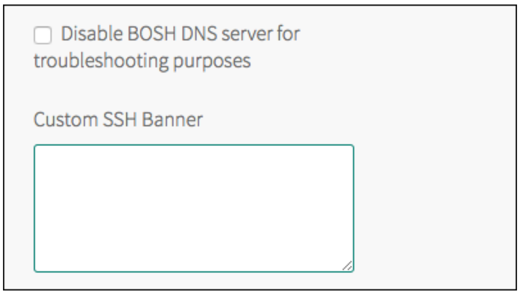
15. (Optional) **Max Threads** sets the maximum number of threads that the Ops Manager Director can run simultaneously. Pivotal recommends that you leave the field blank to use the default value, unless doing so results in rate limiting or errors on your IaaS.

16. (Optional) To add a custom URL for your Ops Manager Director, enter a valid hostname in **Director Hostname**. You can also use this field to configure a load balancer in front of your Ops Manager Director.

17. (Optional) To disable BOSH DNS, select the **Disable BOSH DNS server for troubleshooting purposes** checkbox. For more information about the BOSH DNS service discovery mechanism, see BOSH DNS Service Discovery for Application Containers in the Pivotal Application Service (PAS) Release Notes.

> **Breaking Change**: Do not disable BOSH DNS without consulting Pivotal Support. For more information about disabling BOSH DNS, see Disabling or Opting Out of BOSH DNS in PCF in the Pivotal Knowledge Base.

18. (Optional) To set a custom banner that users see when logging in to the Director using SSH, enter text in the **Custom SSH Banner** field.



19. Click **Save**.

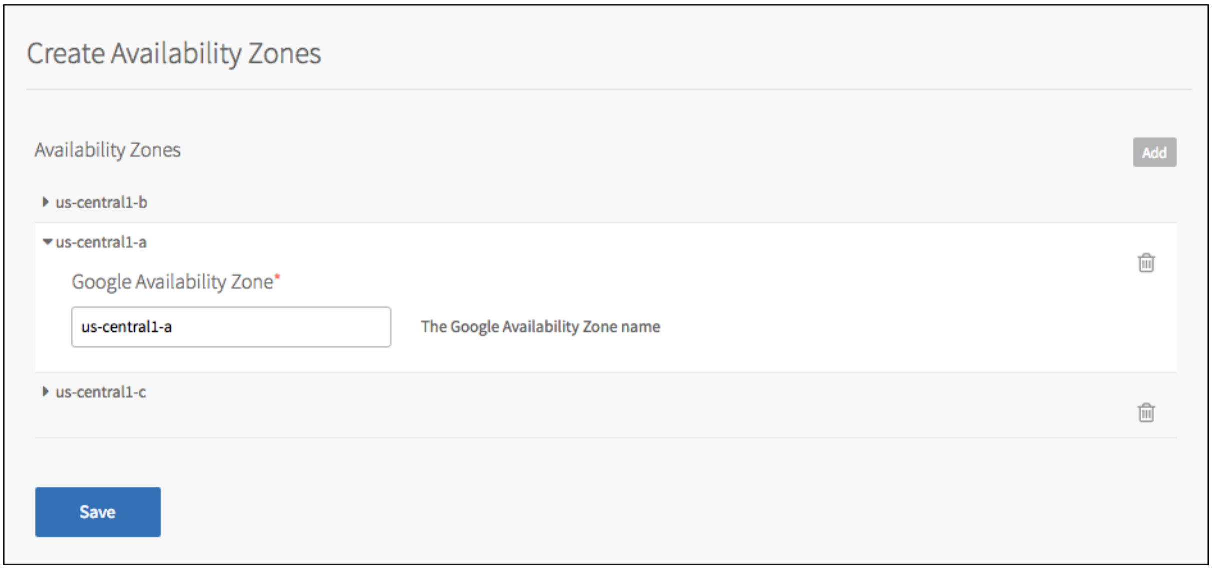
Step 4: Create Availability Zones Page

> **Note**: Pivotal recommends at least three availability zones for a highly available installation of PAS. For an example of a three availability zone deployment, see Reference Architecture for Pivotal Cloud Foundry on GCP.

1. Select **Create Availability Zones**.

2. Click **Add**.

3. For **Google Availability Zone**:
   - Enter one of the zones that you associated to the backend service instance groups of the HTTP(S) Load Balancer. For example, if you are using the `us-central1` region and selected `us-central1-a` as one of the zones for your HTTP(S) Load Balancer instance groups, enter `us-central1-a`.
   - Click **Add**
   - Repeat the above step for all the availability zones that you associated to instance groups in Preparing to Deploy PCF on GCP.



   - Click **Save**.

4. Repeat the above step for all the availability zones you are using in your deployment. When you are done, click **Save**.

Step 5: Create Networks Page

1. Select **Create Networks**.

2. Make sure **Enable ICMP checks** is not selected. GCP routers do not respond to ICMP pings.

3. Use the **Add Network** button to create the following three networks:

> **Note:** To use a shared VPC network, enter the shared VPC host project name before the network name in the format `VPC-PROJECT-NAME/NETWORK-NAME/SUBNET-NAME/REGION-NAME`. For example, `vpc-project/opsmgr/central/us-central1`. For more information, see Configuring a Shared VPC on GCP.

> **Note:** Pivotal recommends using the gateway IP as your default DNS server only if the IP is reachable by all deployed VMs and apps. If you configure Application Security Groups (ASGs) that do not allow apps to reach the gateway IP, for example, your applications will fail to resolve any DNS entries. In this case, consider using other Google-provided DNS servers such as `169.254.169.254` or `8.8.8.8`, as long as these IPs are whitelisted in your configured ASGs.

<table>
<tr><th>Network Name</th><th>Configuration</th></tr>
<tr><td><code>infrastructure</code></td><td>
<table>
<tr><td>Name</td><td><code>infrastructure</code></td></tr>
<tr><td>Service Network</td><td>Leave this checkbox unselected.</td></tr>
<tr><td>Google Network Name</td><td>Enter the name of the infrastructure network you created when preparing your GCP environment. The format is: <code>MY-PCF-virt-net/MY-PCF-subnet-infrastructure-MY-REGION/MY-GCP-REGION</code></td></tr>
<tr><td>CIDR</td><td>Enter the name of the CIDR ending in <code>/26</code> you used when creating the infrastructure subnet in GCP. Example: <code>192.168.101.0/26</code></td></tr>
<tr><td>Reserved IP Ranges</td><td>Enter the first <code>.1</code> through <code>.9</code> addresses from the CIDR. For example, if the CIDR is <code>192.168.101.0/26</code>, enter the range <code>192.168.101.1-192.168.101.9</code>.</td></tr>
</table>
</td></tr>
</table>

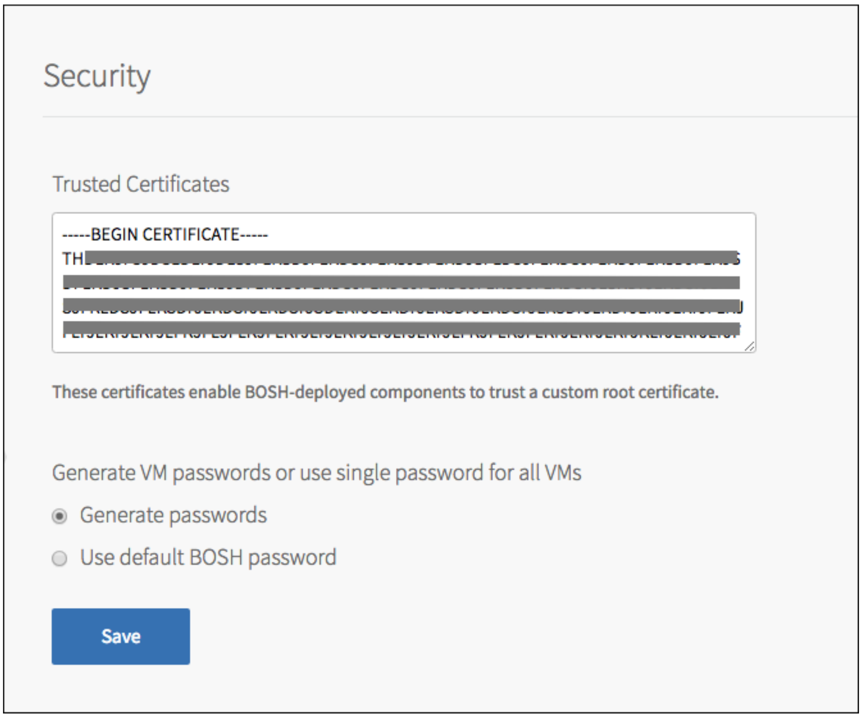
	DNS	Enter the first `.1` address from the CIDR. For example, if the CIDR is `192.168.101.0/26`, enter `192.168.101.1`.
	Gateway	Enter the first `.1` address from the CIDR. For example, if the CIDR is `192.168.101.0/26`, enter `192.168.101.1`.
	Availability Zones	Select all three availability zones.
`ert`	**Name**	`ert`
	Service Network	Leave this checkbox unselected.
	Google Network Name	Enter the name of the PAS network you created when preparing your GCP environment. The format is: `MY-PCF-virt-net/MY-PCF-subnet-ert-MY-REGION/MY-GCP-REGION`
	CIDR	Enter the name of the CIDR ending in `/22` you used when creating the PAS subnet in GCP. Example: `192.168.16.0/22`
	Reserved IP Ranges	Enter the first `.1` through `.9` addresses from the CIDR. For example, if the CIDR is `192.168.16.0/22`, enter the range `192.168.16.1-192.168.16.9`.
	DNS	Enter the first `.1` address from the CIDR. For example, if the CIDR is `192.168.16.0/22`, enter `192.168.16.1`.
	Gateway	Enter the first `.1` address from the CIDR. For example, if the CIDR is `192.168.16.0/22`, enter `192.168.16.1`.
	Availability Zones	Select all three availability zones.
`services-1`	**Name**	`services-1`
	Service Network	Select the checkbox.
	Google Network Name	Enter the name of the services network you created when preparing your GCP environment. The format is: `MY-PCF-virt-net/MY-PCF-subnet-services-MY-REGION/MY-GCP-REGION`
	CIDR	Enter the name of the CIDR ending in `/22` you used when creating the services subnet in GCP. Example: `192.168.20.0/22`
	Reserved IP Ranges	Enter the first `.1` through `.9` addresses from the CIDR. For example, if the CIDR is `192.168.16.0/22`, enter the range `192.168.20.1-192.168.20.9`.
	DNS	Enter the first `.1` address from the CIDR. For example, if the CIDR is `192.168.20.0/22`, enter `192.168.20.1`.
	Gateway	Enter the first `.1` address from the CIDR. For example, if the CIDR is `192.168.20.0/22`, enter `192.168.20.1`.
	Availability Zones	Select all three availability zones.

Step 6: Assign AZs and Networks Page

1. Select **Assign AZs and Networks**.
2. Use the drop-down menu to select a **Singleton Availability Zone**. The Ops Manager Director installs in this Availability Zone.
3. Under **Network**, select the `infrastructure` network for your Ops Manager Director.
4. Click **Save**.

Step 7: Security Page

1. Select **Security**.



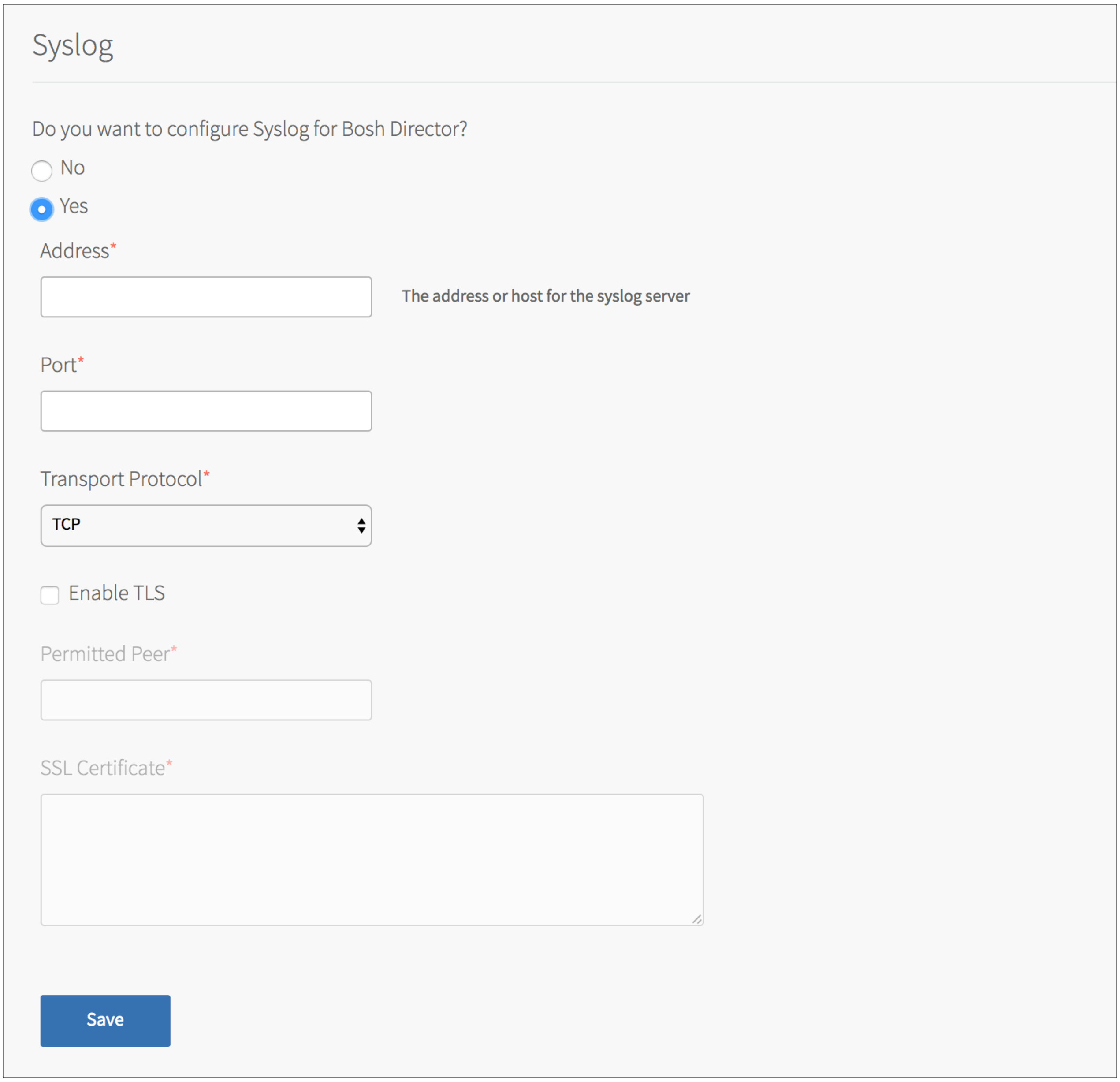
2. In **Trusted Certificates**, enter a custom certificate authority (CA) certificate to insert into your organization's certificate trust chain. This feature enables all BOSH-deployed components in your deployment to trust a custom root certificate.
    - You do not need to enter anything in this field if you are using self-signed certificates.
    - If you want to use Docker Registries for running app instances in Docker containers, enter the certificate for your private Docker Registry in this field. See the Using Docker Registries topic for more information.

3. Choose **Generate passwords** or **Use default BOSH password**. Pivotal recommends that you use the **Generate passwords** option for greater security.

4. Click **Save**. To view your saved Director password, click the **Credentials** tab.

Step 8: Syslog Page

1. Select **Syslog**.



2. (Optional) To send BOSH Director system logs to a remote server, select **Yes**.

3. In the **Address** field, enter the IP address or DNS name for the remote server.

4. In the **Port** field, enter the port number that the remote server listens on.

5. In the **Transport Protocol** dropdown menu, select **TCP**, **UDP**, or **RELP**. This selection determines which transport protocol is used to send the logs to the remote server.

6. (Optional) Mark the **Enable TLS** checkbox to use TLS encryption when sending logs to the remote server.
   - In the **Permitted Peer** field, enter either the name or SHA1 fingerprint of the remote peer.
   - In the **SSL Certificate** field, enter the SSL certificate for the remote server.

7. Click **Save**.

Step 9: Resource Config Page

1. Select **Resource Config**.

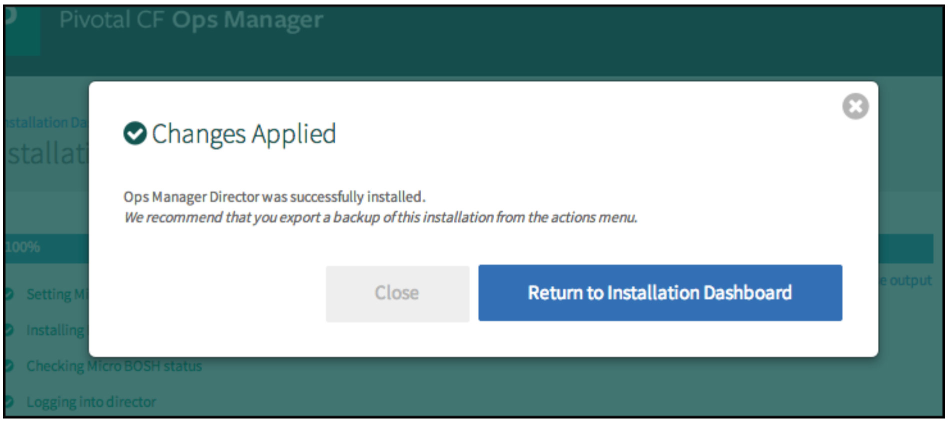
2. Ensure that the **Internet Connected** checkboxes are not selected for any jobs. The checkbox gives VMs a public IP address that enables outbound Internet access. In Preparing to Deploy PCF on GCP, you provisioned a Network Address Translation (NAT) box to provide Internet connectivity to your VMs. For more information about using NAT in GCP, see the GCP documentation.

Step 10: Complete the Ops Manager Director Installation

1. Click the **Installation Dashboard** link to return to the Installation Dashboard.

2. Click **Apply Changes**. If the following ICMP error message appears, return to the Network Config screen, and make sure you have deselected the **Enable ICMP Checks** box. Then click **Apply Changes** again.

3. Ops Manager Director installs. This may take a few moments. When the installation process successfully completes, the **Changes Applied** window appears.



What to Do Next

After you complete this procedure, follow the instructions in the Deploying PAS on GCP topic.

Deploying PAS on GCP

Page last updated:

This topic describes how to install and configure Pivotal Application Service (PAS) on Google Cloud Platform (GCP).

Before beginning this procedure, ensure that you have successfully completed the Configuring Ops Manager Director on GCP topic.

Note: If you plan to install the PCF IPsec add-on, you must do so before installing any other tiles. Pivotal recommends installing IPsec immediately after Ops Manager, and before installing the PAS tile.

Step 1: Download the PAS Tile

1. If you have not already downloaded PAS, log in to Pivotal Network, and click PAS.
2. From the **Releases** drop-down, select the release to install and choose one of the following:
   a. Click PAS to download the PAS `.pivotal` file
   b. Click **PCF Small Footprint Runtime** to download the Small Footprint Runtime `.pivotal` file. For more information, see Getting Started with Small Footprint Runtime.

Step 2: Add PAS to Ops Manager

1. Navigate to the Pivotal Cloud Foundry Operations Manager Installation Dashboard.
2. Click **Import a Product** to add the PAS tile to Ops Manager. This may take a while depending on your connection speed.

   Tip: After you import a tile to Ops Manager, you can view the latest available version of that tile in the Installation Dashboard by enabling the Pivotal Network API. For more information, refer to the Adding and Deleting Products topic.

3. On the left, click the plus icon next to the imported PAS product to add it to the Installation Dashboard.
4. Click the newly added PAS tile in the Installation Dashboard.

Step 3: Assign Availability Zones and Networks

1. Select **Assign AZ and Networks**. These are the Availability Zones that you create when configuring Ops Manager Director.
2. Select an the first Availability Zone under **Place singleton jobs**. Ops Manager runs any job with a single instance in this Availability Zone.
3. Select all Availability Zones under **Balance other jobs**. Ops Manager balances instances of jobs with more than one instance across the Availability Zones that you specify.

   Note: For production deployments, Pivotal recommends at least three Availability Zones for a highly available installation of PAS.

4. From the **Network** drop-down box, choose the `ert` network you created when configuring Ops Manager Director.

5. Click **Save**.

Step 4: Add DNS Records for Your Load Balancers

In this step you redirect queries for your domain to the IP addresses of your load balancers.

1. Locate the static IP addresses of the load balancers you created in Preparing to Deploy PCF on GCP:
    - An HTTP(S) load balancer named `MY-PCF-global-pcf`
    - A TCP load balancer for WebSockets named `MY-PCF-wss-logs`
    - A TCP load balancer for SSH access to applications named `MY-PCF-ssh-proxy`
    - A TCP load balancer for the TCP router named `MY-PCF-cf-tcp-lb` if you plan on enabling the TCP routing feature

    > **Note**: You can locate the static IP address of each load balancer by clicking its name under **Networks > Load balancing** in the GCP Console.

2. Log in to the DNS registrar that hosts your domain. Examples of DNS registrars include Network Solutions, GoDaddy, and Register.com.

3. Create **A records** with your DNS registrar that map domain names to the public static IP addresses of the load balancers located above:

If your DNS entry is:	Set to the public IP of this load balancer:	Required	Example
*.YOURSYSTEMDOMAIN	`MY-PCF-global-pcf`	Yes	*.system.example.com
*.YOURAPPSDOMAIN	`MY-PCF-global-pcf`	Yes	*.apps.example.com
doppler.YOURSYSTEMDOMAIN	`MY-PCF-wss-logs`	Yes	doppler.system.example.com
loggregator.YOURSYSTEMDOMAIN	`MY-PCF-wss-logs`	Yes	loggregator.system.example.com
ssh.YOURSYSTEMDOMAIN	`MY-PCF-ssh-proxy`	Yes, to allow SSH access to apps	ssh.system.example.com
tcp.YOURDOMAIN	`MY-PCF-cf-tcp-lb`	No, only set up if you have enabled the TCP routing feature	tcp.example.com

4. Save changes within the web interface of your DNS registrar.

5. In a terminal window, run the following `dig` command to confirm that you created your A record successfully:

```
dig xyz.EXAMPLE.COM
```

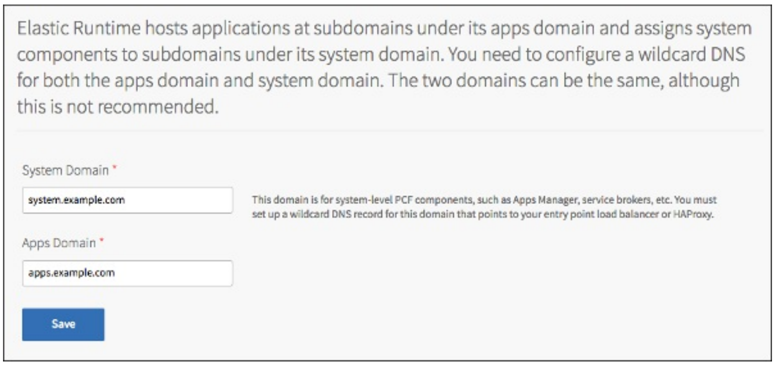
    You should see the A record that you just created:

```
;; ANSWER SECTION:
xyz.EXAMPLE.COM.    1767    IN  A 203.0.113.1
```

> **Note**: You **must** complete this step before proceeding to Cloud Controller configuration. A difficult-to-resolve problem can occur if the wildcard domain is improperly cached before the A record is registered.

Step 5: Configure Domains

1. Select **Domains**.



2. Enter the system and application domains that you created when [preparing your GCP environment](#).
   - The **System Domain** defines your target when you push apps to PAS.
   - The **Apps Domain** defines where PAS serves your apps.

> **Note**: Pivotal recommends that you use the same domain name but different subdomain names for your system and app domains. For example, use `system.example.com` for your system domain, and `apps.example.com` for your apps domain.

3. Click **Save**.

Step 6: Configure Networking

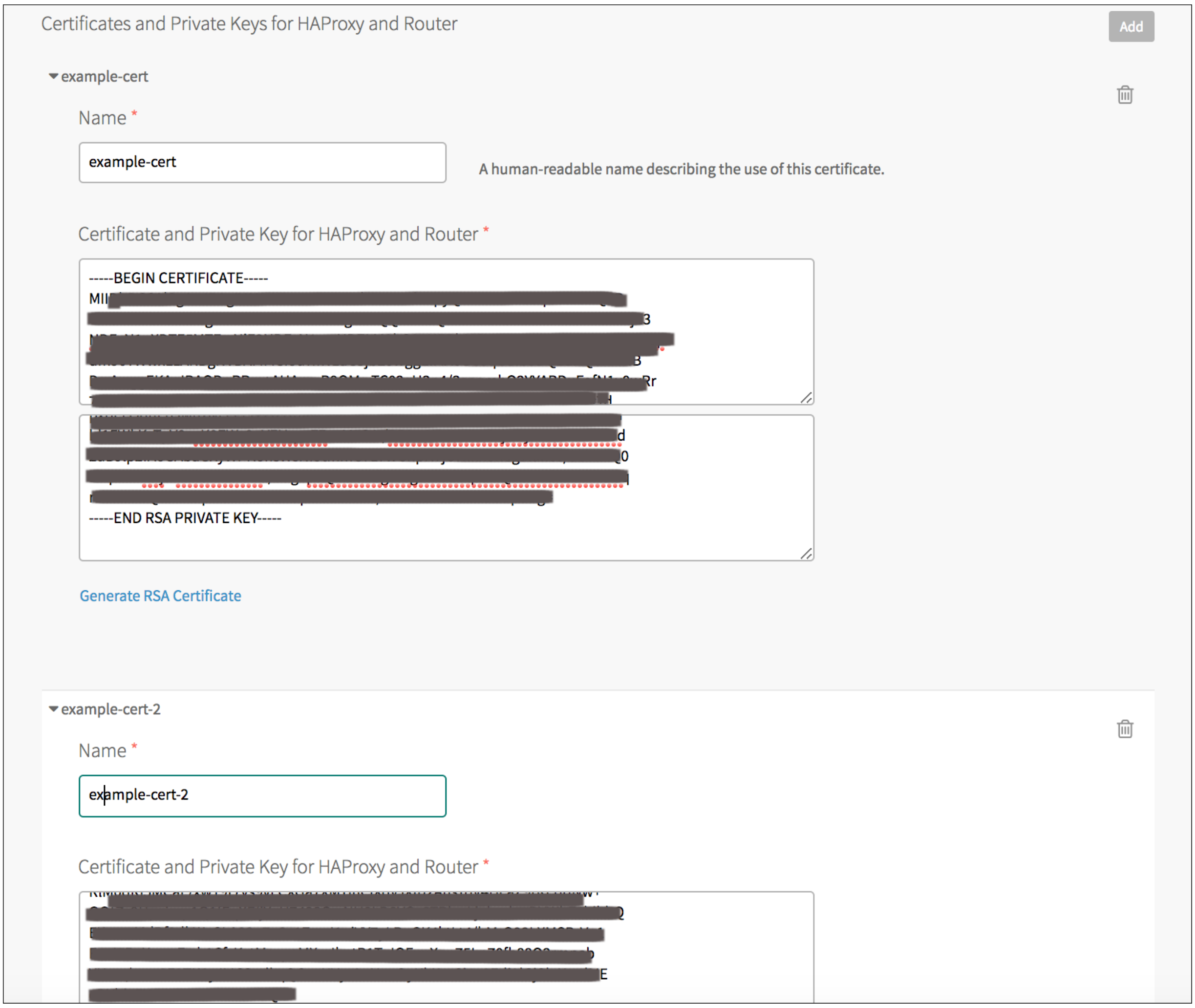
1. Select **Networking**.

2. Leave the **Router IPs**, **SSH Proxy IPs**, **HAProxy IPs**, and **TCP Router IPs** fields blank. You do not need to complete these fields when deploying PCF to GCP.

> **Note**: You specify load balancers in the **Resource Config** section of PAS later on in the installation process. See the [Configure Load Balancers](#) section of this topic for more information.

3. Under **Certificates and Private Key for HAProxy and Router**, you must provide at least one **Certificate and Private Key** name and certificate keypair for HAProxy and Gorouter. The HAProxy and Gorouter are enabled to receive TLS communication by default. You can configure multiple certificates for HAProxy and Gorouter.

   a. Click the **Add** button to add a name for the certificate chain and its private keypair. This certificate is the default used by Gorouter and HAProxy.

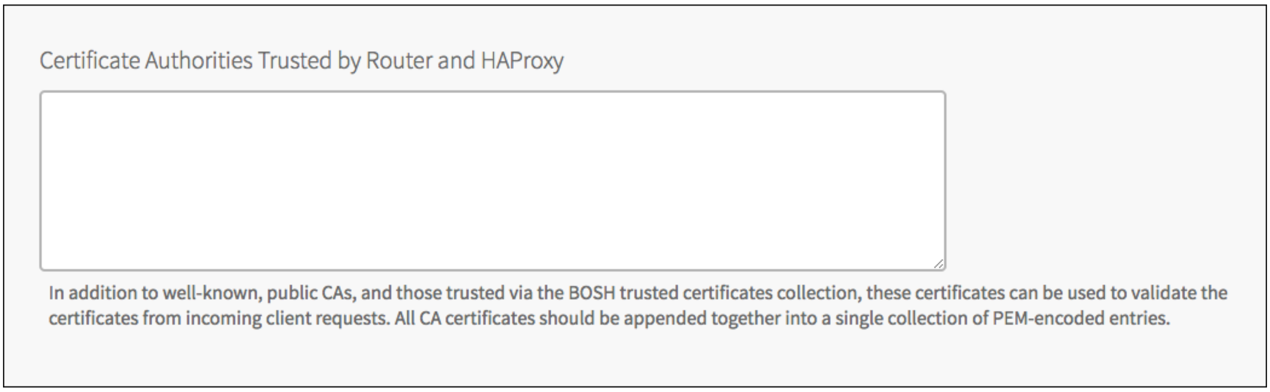


You can either provide a certificate signed by a Certificate Authority (CA) or click on the **Generate RSA Certificate** link to generate a self-signed certificate in Ops Manager.

b. If you want to configure multiple certificates for HAProxy and Gorouter, click the **Add** button and fill in the appropriate fields for each additional certificate keypair.

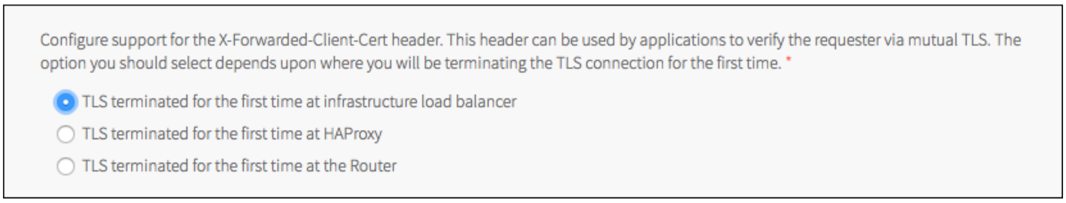
For details about generating certificates in Ops Manager for your wildcard system domains, see the Providing a Certificate for Your SSL/TLS Termination Point topic.

4. (Optional) When validating client requests using mutual TLS, the Gorouter trusts multiple certificate authorities (CAs) by default. If you want to configure the Gorouter and HAProxy to trust additional CAs, enter your CA certificates under **Certificate Authorities Trusted by Router and HAProxy**. All CA certificates should be appended together into a single collection of PEM-encoded entries.



5. In the **Minimum version of TLS supported by HAProxy and Router** field, select the minimum version of TLS to use in HAProxy and Router communications. HAProxy and Router use TLS v1.2 by default. If you need to accommodate clients that use an older version of TLS, select a lower minimum version. For a list of TLS ciphers supported by the Gorouter, see Securing Traffic into Cloud Foundry.

6. Under **Configure support for the X-Forwarded-Client-Cert header**, configure PCF handles `x-forwarded-client-cert` (XFCC) HTTP headers based on where TLS is terminated for the first time in your deployment.

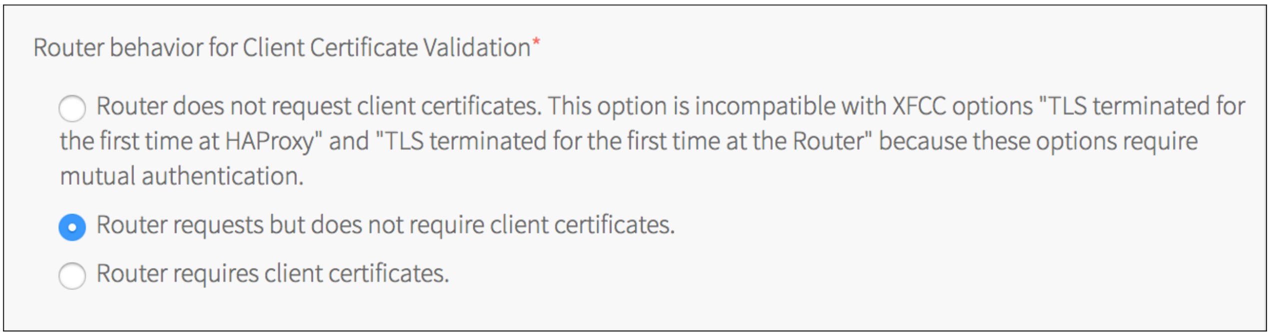


The following table indicates which option to choose based on your deployment layout.

If your deployment is configured as follows:	Then select the following option:	Additional notes:
◦ The Load Balancer is terminating TLS, and ◦ Load balancer is configured to put the client certificate from a mutual authentication TLS handshake into the X-Forwarded-Client-Cert HTTP header	TLS terminated for the first time at infrastructure load balancer (default).	Both HAProxy and the Gorouter forward the XFCC header when included in the request.
◦ The Load Balancer is configured to pass through the TLS handshake via TCP to the instances of HAProxy, and ◦ HAProxy instance count is > 0	TLS terminated for the first time at HAProxy.	HAProxy sets the XFCC header with the client certificate received in the TLS handshake. The Gorouter forwards the header. **Breaking Change**: In the **Router behavior for Client Certificates** field, you cannot select the **Router does not request client certificates** option.
◦ The Load Balancer is configured to pass through the TLS handshake via TCP to instances of the Gorouter	TLS terminated for the first time at the Gorouter.	The Gorouter strips the XFCC header if it is included in the request and forwards the client certificate received in the TLS handshake in a new XFCC header. If you have deployed instances of HAProxy, app traffic bypasses those instances in this configuration. If you have also configured your load balancer to route requests for ssh directly to the Diego Brain, consider reducing HAProxy instances to 0. **Breaking Change**: In the **Router behavior for Client Certificates** field, you cannot select the **Router does not request client certificates** option.

For a description of the behavior of each configuration option, see Forward Client Certificate to Applications.

7. To configure Gorouter behavior for handling client certificates, select one of the following options in the **Router behavior for Client Certificate Validation** field.



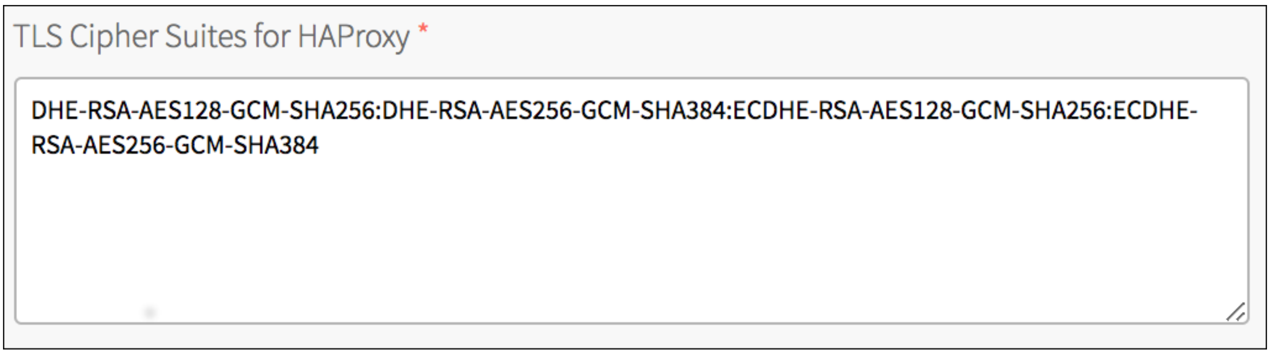
- **Router does not request client certificates.** This option is incompatible with the XFCC configuration options **TLS terminated for the first time at HAProxy** and **TLS terminated for the first time at the Router** in PAS because these options require mutual authentication. As client certificates are not requested, client will not provide them, and thus validation of client certificates will not occur.
- **Router requests but does not require client certificates.** Gorouter requests client certificates in TLS handshakes, validates them when presented, but does not require them. This is the default configuration.
- **Router requires client certificates.** Gorouter validates that the client certificate is signed by a Certificate Authority that Gorouter trusts. If Gorouter cannot validate the client certificate, the TLS handshake fails.

8. In the **TLS Cipher Suites for Router** field, specify the TLS cipher suites to use for TLS handshakes between the Gorouter and downstream clients like load balancers or HAProxy. Use an ordered, colon-delimited list of Golang-supported TLS cipher suites in the OpenSSL format. The recommended setting is `ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384`. Operators should verify that the ciphers are supported by any clients or downstream components that will initiate TLS handshakes with the Gorouter. For a list of TLS ciphers supported by the Gorouter, see Securing Traffic into Cloud Foundry.

Verify that whatever client is participating in the TLS handshake with the Gorouter has at least one cipher suite in common with the Gorouter.

> **Note:** Specify cipher suites that are supported by the versions configured in the **Minimum version of TLS supported by HAProxy and Router** field.

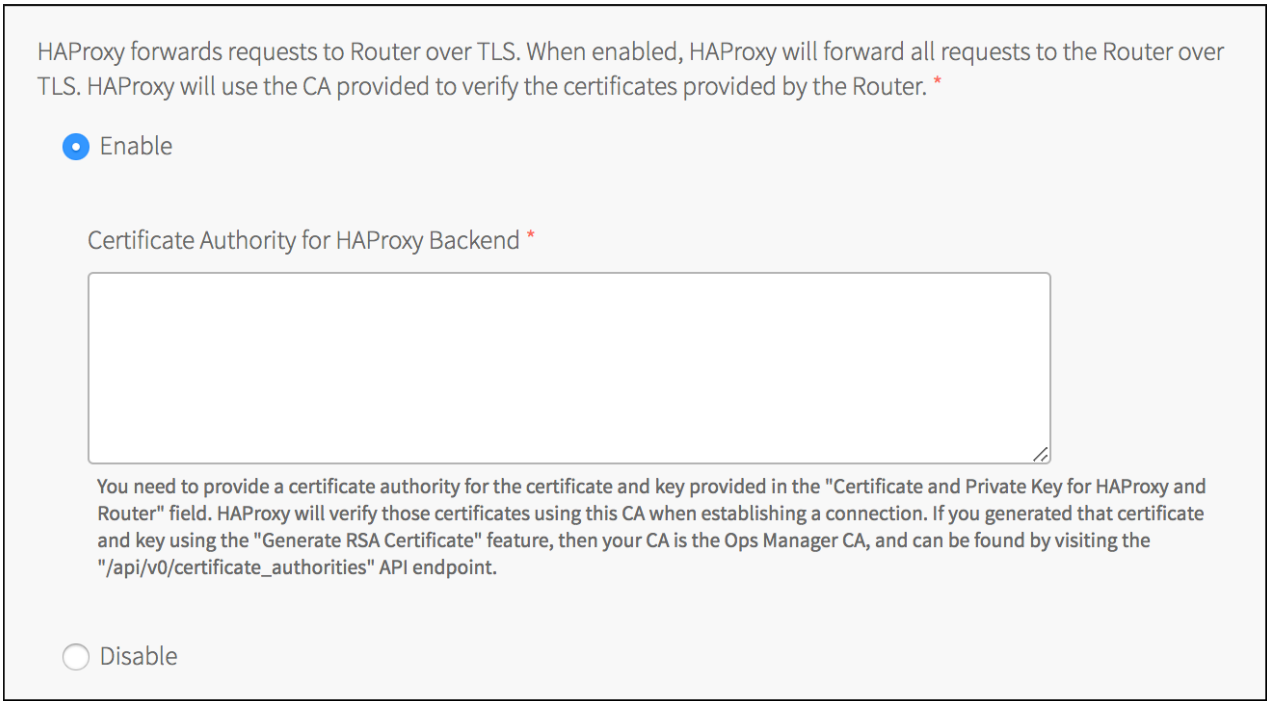
9. In the **TLS Cipher Suites for HAProxy** field, specify the TLS cipher suites to use in HAProxy for TLS handshakes between HAProxy and its clients such as load balancers and Gorouter. Use an ordered, colon-delimited list of TLS cipher suites in the OpenSSL format. The recommended setting: `DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384`
Operators should verify that the ciphers are supported by any clients or downstream components that will initiate TLS handshakes with the HAProxy.



Verify that whatever clients are participating in the TLS handshake with HAProxy have at least one cipher suite in common with HAProxy.

> **Note:** Specify cipher suites that are supported by the versions configured in the **Minimum version of TLS supported by HAProxy and Router** field.

10. Under **HAProxy forwards requests to Router over TLS**, select **Enable** or **Disable** based on your deployment layout.



- **Enable HAProxy forwarding of requests to Router over TLS**

If you want to:	Encrypt communication between HAProxy and the Gorouter
Then configure the following:	1. Leave **Enable** selected. 2. In the **Certificate Authority for HAProxy Backend** field, specify the Certificate Authority (CA) that signed the certificate you configured in the **Certificate and Private Key for HAProxy and Router** field. **Note**: If you used the **Generate RSA Certificate** link to generate a self-signed certificate, then the CA to specify is the Ops Manager CA, which you can locate at the CA endpoint in the Ops Manager API. 3. Make sure that Gorouter and HAPRoxy have TLS cipher suites in common in the **TLS Cipher Suites for Router** and **TLS Cipher Suites for HAProxy** fields.
See also:	◦ Terminating SSL/TLS at the Load Balancer and Gorouter ◦ Providing a Certificate for Your SSL/TLS Termination Point ◦ Using the Ops Manager API

- **Disable HAProxy forwarding of requests to Router over TLS**

If you want to:	Use non-encrypted communication between HAProxy and Gorouter, or you are not using HAProxy
Then configure the following:	1. Select **Disable**. 2. If you are not using HAProxy, set the number of HAProxy job instances to `0` on the **Resource Config** page. See Disable Unused Resources.
See also:	◦ Terminating SSL/TLS at the Gorouter Only ◦ Terminating SSL/TLS at the Load Balancer Only

11. If you are not using SSL encryption or if you are using self-signed certificates, select **Disable SSL certificate verification for this environment**. Selecting this checkbox also disables SSL verification for route services.

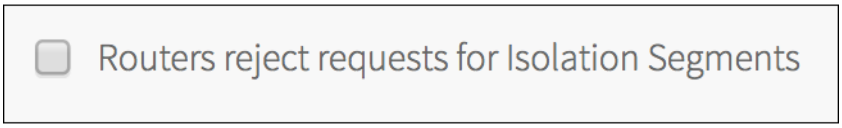
> **Note**: For production deployments, Pivotal does not recommend disabling SSL certificate verification.

12. (Optional) If you want HAProxy or the Gorouter to reject any HTTP (non-encrypted) traffic, select the **Disable HTTP on HAProxy and Gorouter** checkbox. When selected, HAProxy and Gorouter will not listen on port 80.

13. (Optional) Select the **Disable insecure cookies on the Router** checkbox to set the secure flag for cookies generated by the router.

14. (Optional) To disable the addition of Zipkin tracing headers on the Gorouter, deselect the **Enable Zipkin tracing headers on the router** checkbox. Zipkin tracing headers are enabled by default. For more information about using Zipkin trace logging headers, see Zipkin Tracing in HTTP Headers.

15. (Optional) To stop the Router from writing access logs to local disk, deselect the **Enable Router to write access logs locally** checkbox. You should consider disabling this checkbox for high traffic deployments since logs may not be rotated fast enough and can fill up the disk.

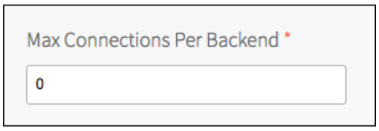
16. By default, the PAS routers handle traffic for applications deployed to an isolation segment created by the PCF Isolation Segment tile. To configure the PAS routers to reject requests for applications within isolation segments, select the **Routers reject requests for Isolation Segments** checkbox.


Do not enable this option without deploying routers for each isolation segment. See the following topics for more information:
    - Installing PCF Isolation Segment
    - Sharding Routers for Isolation Segments.

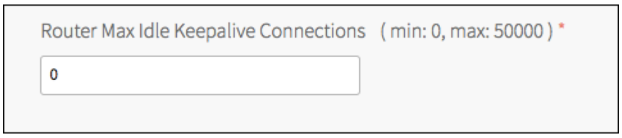
17. In the **Choose whether or not to enable route services** section, choose either **Enable route services** or **Disable route services**. Route services are a class of marketplace services that perform filtering or content transformation on application requests and responses. See the Route Services topic for details.

18. (Optional) If you want to limit the number of app connections to the backend, enter a value in the **Max Connections Per Backend** field. You can use this field to prevent a poorly behaving app from all the connections and impacting other apps.

    To choose a value for this field, review the peak concurrent connections received by instances of the most popular apps in your deployment. You can determine the number of concurrent connections for an app from the `httpStartStop` event metrics emitted for each app request.

    If your deployment uses PCF Metrics, you can also obtain this peak concurrent connection information from Network Metrics. The default value is


`500`.

19. Enter a value for **Router Max Idle Keepalive Connections**. See Considerations for Configuring max_idle_connections.


20. (Optional) To accommodate larger uploads over connections with high latency, increase the number of seconds in the **Router Timeout to Backends** field.

21. (Optional) Use the **Frontend Idle Timeout for Gorouter and HAProxy** field to help prevent connections from your load balancer to Gorouter or HAProxy from being closed prematurely. The value you enter sets the duration, in seconds, that Gorouter or HAProxy maintains an idle open connection from a load balancer that supports keep-alive.

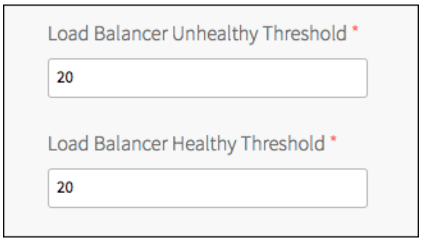
    In general, set the value higher than your load balancer's backend idle timeout to avoid the race condition where the load balancer sends a request before it discovers that Gorouter or HAProxy has closed the connection.

    See the following table for specific guidance and exceptions to this rule:

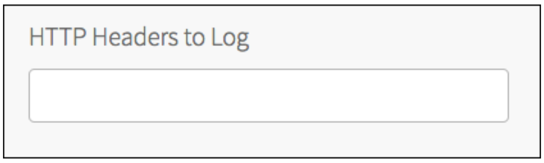
IaaS	Guidance
AWS	AWS ELB has a default timeout of 60 seconds, so Pivotal recommends a value greater than `60`.
Azure	By default, Azure load balancer times out at 240 seconds without sending a TCP RST to clients, so as an exception, Pivotal recommends a value lower than `240` to force the load balancer to send the TCP RST.
GCP	GCP has a default timeout of 600 seconds, so Pivotal recommends a value greater than `600`.
Other	Set the timeout value to be greater than that of the load balancer's backend idle timeout.

22. (Optional) Increase the value of **Load Balancer Unhealthy Threshold** to specify the amount of time, in seconds, that the router continues to accept connections before shutting down. During this period, healthchecks may report the router as unhealthy, which causes load balancers to failover to other routers. Set this value to an amount greater than or equal to the maximum time it takes your load balancer to consider a router instance unhealthy, given contiguous failed healthchecks.

23. (Optional) Modify the value of **Load Balancer Healthy Threshold**. This field specifies the amount of time, in seconds, to wait until declaring the Router instance started. This allows an external load balancer time to register the Router instance as healthy.

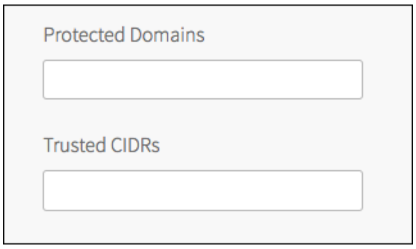


24. (Optional) If app developers in your organization want certain HTTP headers to appear in their app logs with information from the Gorouter, specify them in the **HTTP Headers to Log** field. For example, to support app developers that deploy Spring apps to PCF, you can enter [Spring-specific HTTP headers](#).

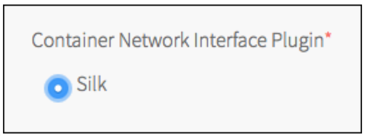


25. If you expect requests larger than the default maximum of 16 Kbytes, enter a new value (in bytes) for **HAProxy Request Max Buffer Size**. You may need to do this, for example, to support apps that embed a large cookie or query string values in headers.

26. If your PCF deployment uses HAProxy and you want it to receive traffic only from specific sources, use the following fields:
    - **Protected Domains:** Enter a comma-separated list of domains from which PCF can receive traffic.
    - **Trusted CIDRs:** Optionally, enter a space-separated list of CIDRs to limit which IP addresses from the **Protected Domains** can send traffic to PCF.



27. The **Loggregator Port** defaults to `443` if left blank. Enter a new value to override the default.



28. For **Container Network Plugin Interface**, ensure **Silk** is selected and review the following fields:

> **Note**: The **External** option exists to support NSX-T integration for vSphere deployments.

   a. (Optional) You can change the value in the **Applications Network Maximum Transmission Unit (MTU)** field. Pivotal recommends setting the MTU value for your application network to `1454`. Some configurations, such as networks that use GRE tunnels, may require a smaller MTU value.
   b. (Optional) Enter an IP range for the overlay network in the **Overlay Subnet** box. If you do not set a custom range, Ops Manager uses `10.255.0.0/16`.

   > **WARNING**: The overlay network IP range must not conflict with any other IP addresses in your network.

   c. Enter a UDP port number in the **VXLAN Tunnel Endpoint Port** box. If you do not set a custom port, Ops Manager uses 4789.
   d. For **Denied logging interval**, set the per-second rate limit for packets blocked by either a container-specific [networking policy](#) or by

[Application Security Group](#) rules applied across the space, org, or deployment. This field defaults to `1`.

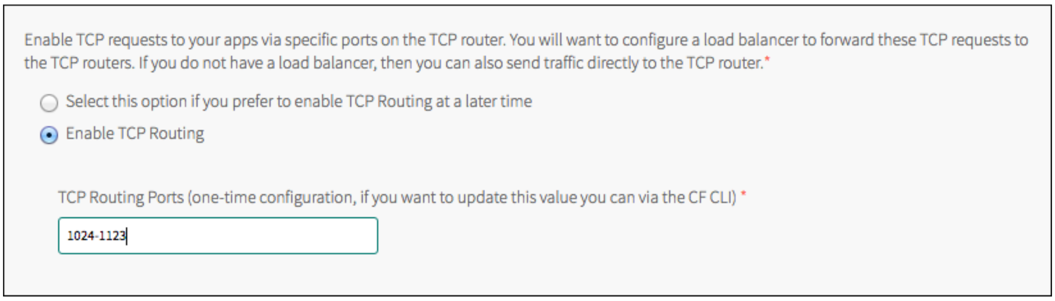
e. For **UDP logging interval**, set the per-second rate limit for UDP packets sent and received. This field defaults to `100`.
f. To enable logging for app traffic, select **Log traffic for all accepted/denied application packets**. See Manage Logging for Container-to-Container Networking for more information.

29.
(Optional) TCP Routing is disabled by default. To enable this feature, perform the following steps:

a. Select **Enable TCP Routing**.
b. In **TCP Routing Ports**, enter a range of ports to be allocated for TCP Routes.

For each TCP route you want to support, you must reserve a range of ports. This is the same range of ports you configured your load balancer with in the Pre-Deployment Steps, unless you configured DNS to resolve the TCP domain name to the TCP router directly.

The **TCP Routing Ports** field accepts a comma-delimited list of individual ports and ranges, for example `1024-1099,30000,60000-60099`. Configuration of this field is only applied on the first deploy, and update updates to the port range are made using the cf CLI. For details about modifying the port range, see the Router Groups topic.



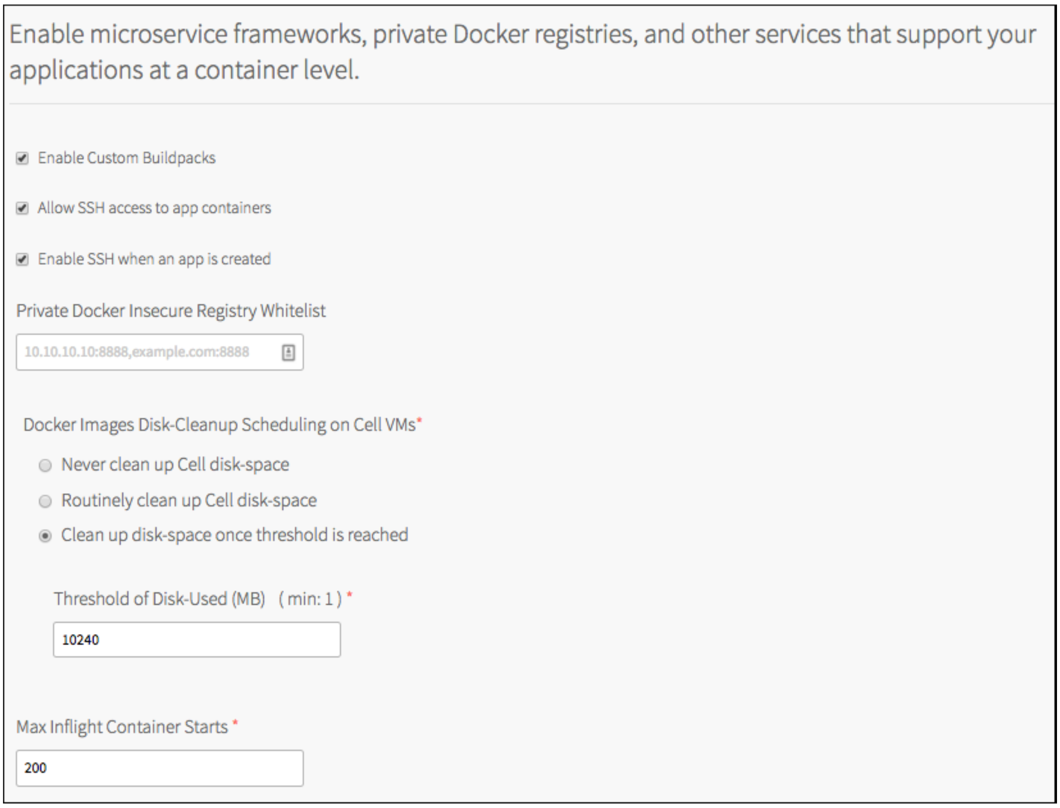
c. For GCP, you also need to specify the name of a GCP TCP load balancer in the **LOAD BALANCER** column of TCP Router job of the **Resource Config** screen. You configure this later on in PAS. See Configure Load Balancers section of this topic.

30. (Optional) To disable TCP routing, click **Select this option if you prefer to enable TCP Routing at a later time** For more information, see the Configuring TCP Routing in PAS topic.

31. Click **Save**.

Step 7: Configure Application Containers

1. Select **Application Containers**.



2. The **Enable Custom Buildpacks** checkbox governs the ability to pass a custom buildpack URL to the `-b` option of the `cf push` command. By default, this ability is enabled, letting developers use custom buildpacks when deploying apps. Disable this option by disabling the checkbox. For more information about custom buildpacks, refer to the buildpacks section of the PCF documentation.

3. The **Allow SSH access to app containers** checkbox controls SSH access to application instances. Enable the checkbox to permit SSH access across your deployment, and disable it to prevent all SSH access. See the Application SSH Overview topic for information about SSH access permissions at the space and app scope.

4. If you want enable SSH access for new apps by default in spaces that allow SSH, select **Enable SSH when an app is created**. If you deselect the checkbox, developers can still enable SSH after pushing their apps by running `cf enable-ssh APP-NAME`.

5. You can configure Pivotal Application Service (PAS) to run app instances in Docker containers by supplying their IP address range(s) in the **Private Docker Insecure Registry Whitelist** textbox. See the Using Docker Registries topic for more information.

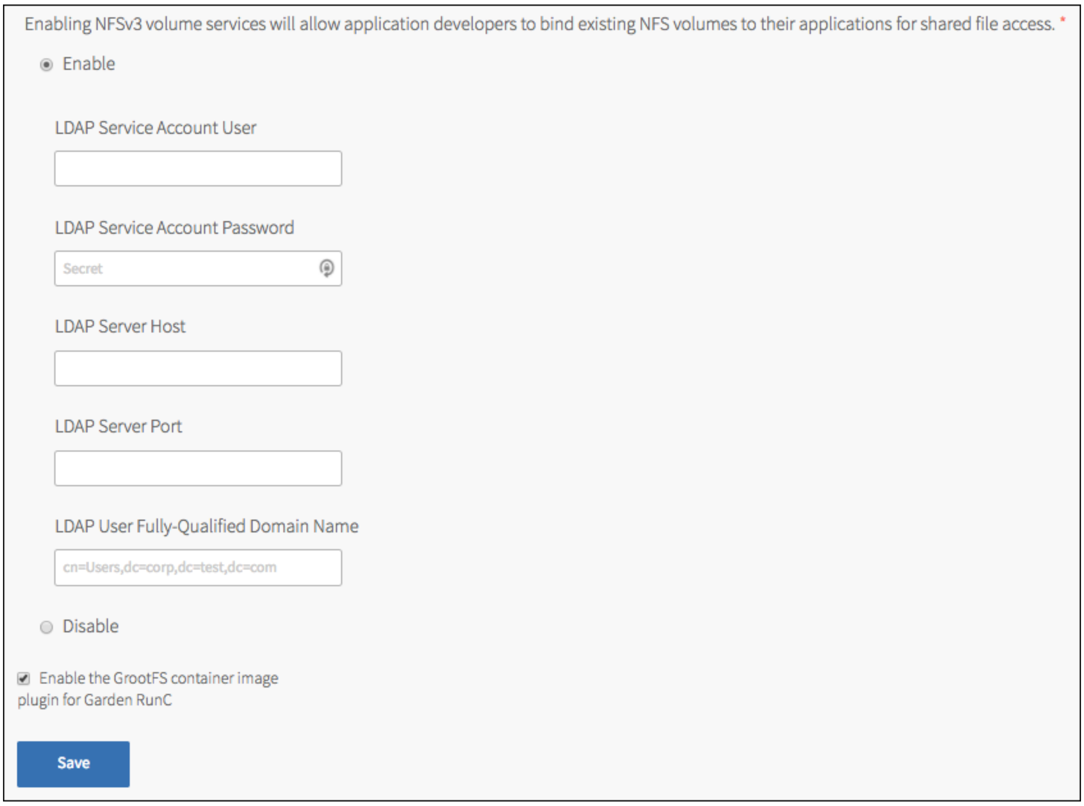
6. Select your preference for **Docker Images Disk-Cleanup Scheduling on Cell VMs**. If you choose **Clean up disk-space once threshold is reached**, enter a **Threshold of Disk-Used** in megabytes. For more information about the configuration options and how to configure a threshold, see Configuring Docker Images Disk-Cleanup Scheduling.

7. Enter a number in the **Max Inflight Container Starts** textbox. This number configures the maximum number of started instances across your deployment's Diego Cells. For more information about this feature, see Setting a Maximum Number of Started Containers.

8. Under **Enabling NFSv3 volume services**, select **Enable** or **Disable**. NFS volume services allow application developers to bind existing NFS volumes to their applications for shared file access. For more information, see the Enabling NFS Volume Services topic.

   > **Note**: In a clean install, NFSv3 volume services is enabled by default. In an upgrade, NFSv3 volume services is set to the same setting as it was in the previous deployment.

9. (Optional) To configure LDAP for NFSv3 volume services, perform the following steps:



- For **LDAP Service Account User**, enter the username of the service account in LDAP that will manage volume services.
- For **LDAP Service Account Password**, enter the password for the service account.
- For **LDAP Server Host**, enter the hostname or IP address of the LDAP server.
- For **LDAP Server Port**, enter the LDAP server port number. If you do not specify a port number, Ops Manager uses 389.
- For **LDAP Server Protocol**, enter the server protocol. If you do not specify a protocol, Ops Manager uses TCP.
- For **LDAP User Fully-Qualified Domain Name**, enter the fully qualified path to the LDAP service account. For example, if you have a service account named `volume-services` that belongs to organizational units (OU) named `service-accounts` and `my-company`, and your domain is named `domain`, the fully qualified path looks like the following:

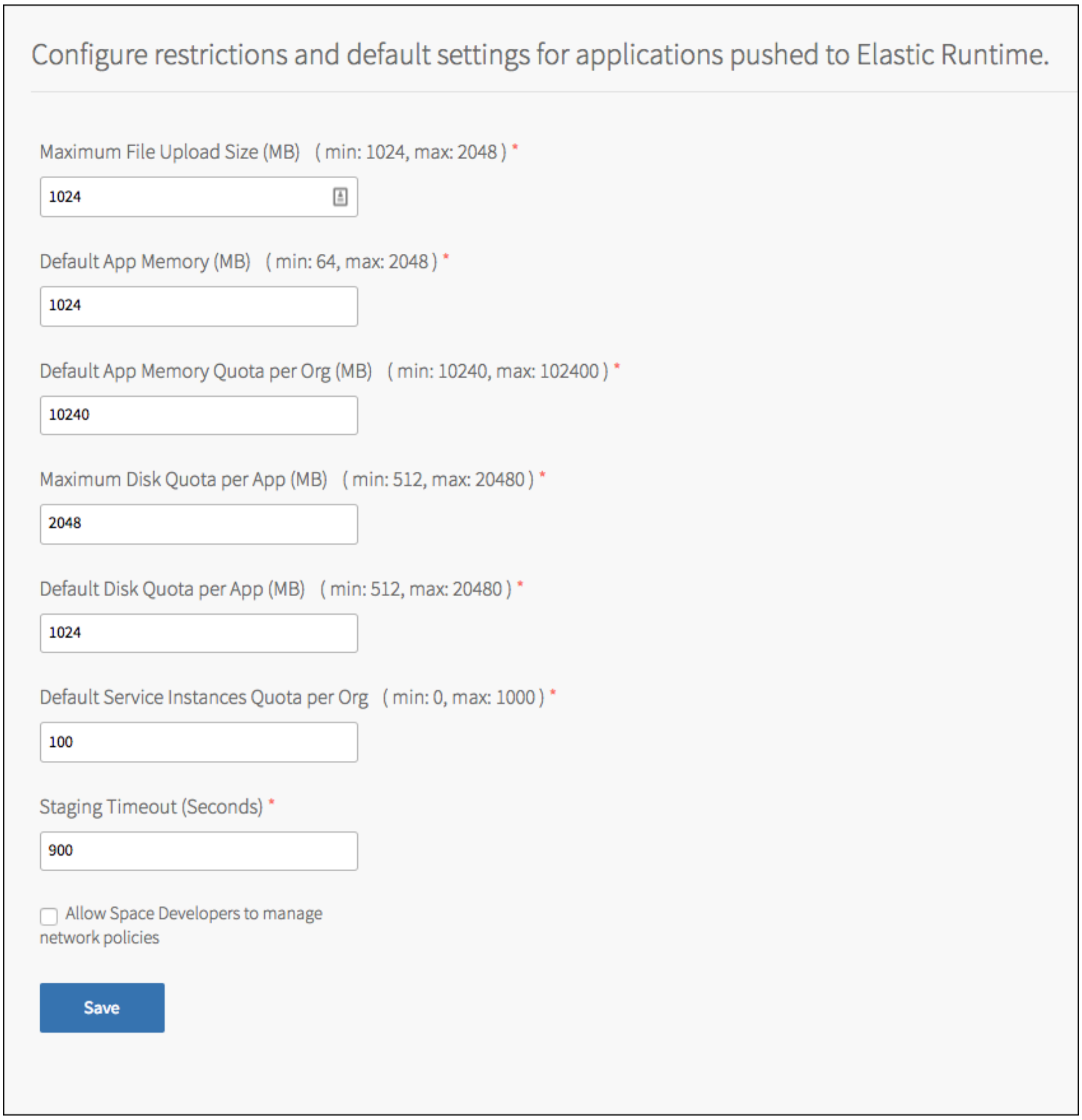
  ```
  CN=volume-services,OU=service-accounts,OU=my-company,DC=domain,DC=com
  ```

10. By default, PAS manages container images using the GrootFS plugin for Garden-runC. If you experience issues with GrootFS, you can disable the plugin and use the image plugin built into Garden-runC.

11. Click **Save**.

Step 8: Configure Application Developer Controls

1. Select **Application Developer Controls**.



2. Enter the **Maximum File Upload Size (MB)**. This is the maximum size of an application upload.

3. Enter the **Default App Memory (MB)**. This is the amount of RAM allocated by default to a newly pushed application if no value is specified with the cf CLI.

4. Enter the **Default App Memory Quota per Org**. This is the default memory limit for all applications in an org. The specified limit only applies to the first installation of PAS. After the initial installation, operators can use the cf CLI to change the default value.

5. Enter the **Maximum Disk Quota per App (MB)**. This is the maximum amount of disk allowed per application.

> **Note**: If you allow developers to push large applications, PAS may have trouble placing them on Cells. Additionally, in the event of a system upgrade or an outage that causes a rolling deploy, larger applications may not successfully re-deploy if there is insufficient disk capacity. Monitor your deployment to ensure your Cells have sufficient disk to run your applications.

6. Enter the **Default Disk Quota per App (MB)**. This is the amount of disk allocated by default to a newly pushed application if no value is specified with the cf CLI.

7. Enter the **Default Service Instances Quota per Org**. The specified limit only applies to the first installation of PAS. After the initial installation, operators can use the cf CLI to change the default value .

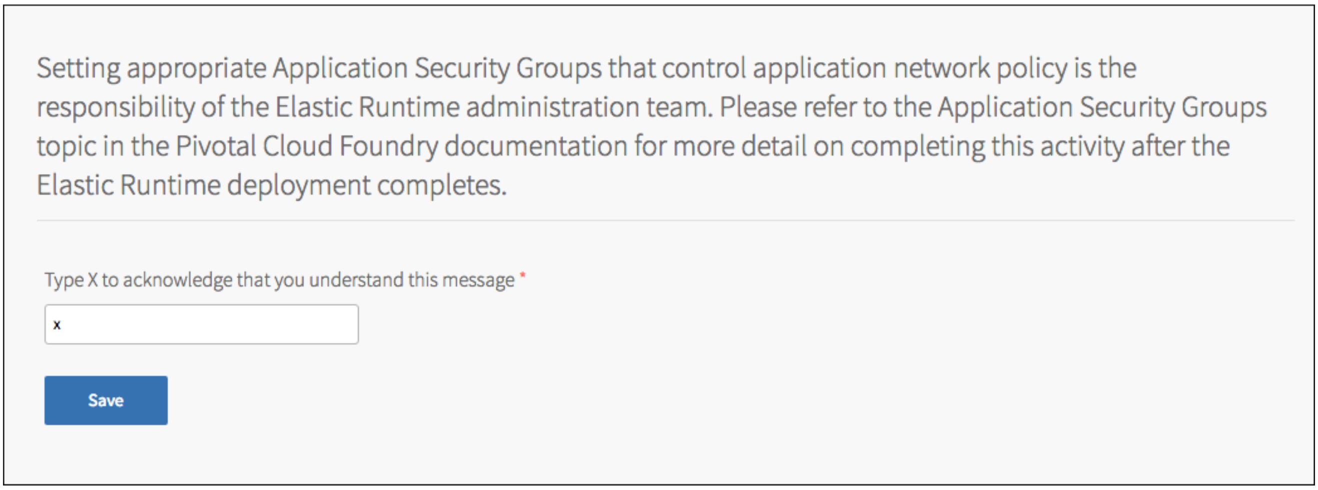
8. Enter the **Staging Timeout (Seconds)**. When you stage an application droplet with the Cloud Controller, the server times out after the number of seconds you specify in this field.

9. Select the **Allow Space Developers to manage network policies** checkbox to permit developers to manage their own network policies for their applications.

10. Click **Save**.

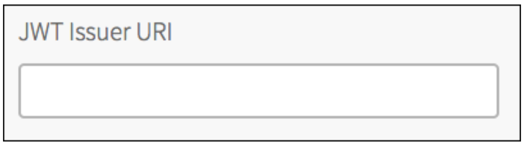
Step 9: Review Application Security Groups

Setting appropriate Application Security Groups is critical for a secure deployment. Type `X` in the box to acknowledge that once the Pivotal Application Service (PAS) deployment completes, you will review and set the appropriate application security groups. See Restricting App Access to Internal PCF Components for instructions.



Step 10: Configure UAA

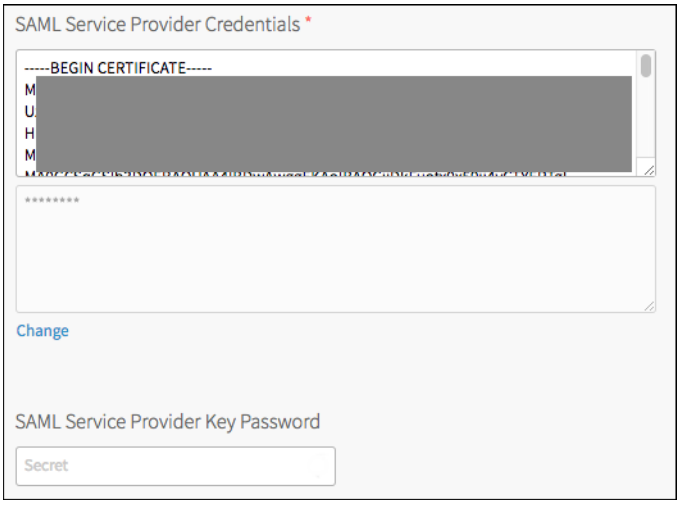
1. Select **UAA**.
2. (Optional) Under **JWT Issuer URI**, enter the URI that UAA uses as the issuer when generating tokens.



3. Under **SAML Service Provider Credentials**, enter a certificate and private key to be used by UAA as a SAML Service Provider for signing outgoing SAML authentication requests. You can provide an existing certificate and private key from your trusted Certificate Authority or generate a self-signed certificate. The following domain must be associated with the certificate: `*.login.YOUR-SYSTEM-DOMAIN`.

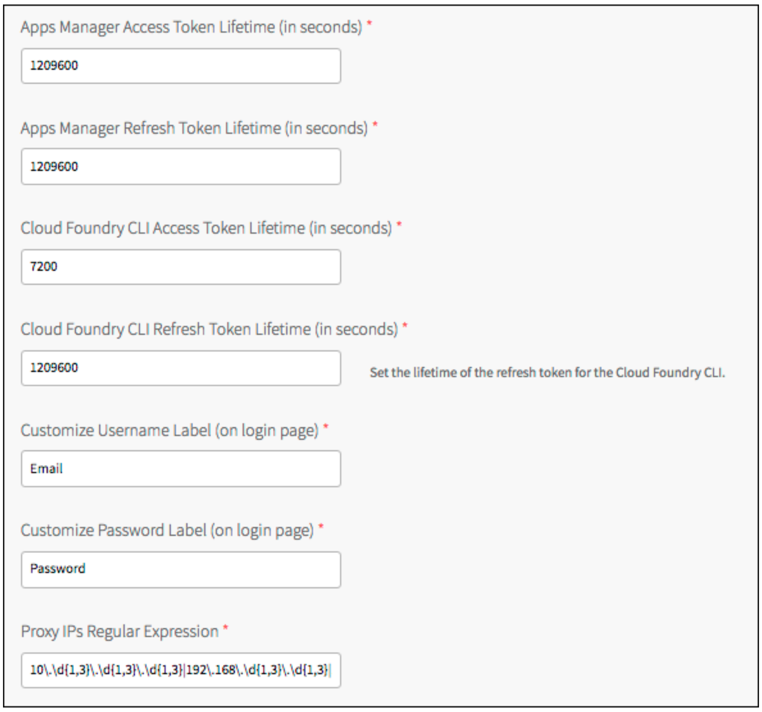
> **Note**: The Pivotal Single Sign-On Service and Pivotal Spring Cloud Services tiles require the `*.login.YOUR-SYSTEM-DOMAIN`.

4. If the private key specified under **Service Provider Credentials** is password-protected, enter the password under **SAML Service Provider Key Password**.



5. For **Signature Algorithm**, choose an algorithm from the dropdown menu to use for signed requests and assertions. The default value is `SHA256`.

6. (Optional) In the **Apps Manager Access Token Lifetime**, **Apps Manager Refresh Token Lifetime**, **Cloud Foundry CLI Access Token Lifetime**, and **Cloud Foundry CLI Refresh Token Lifetime** fields, change the lifetimes of tokens granted for Apps Manager and Cloud Foundry Command Line Interface (cf CLI) login access and refresh. Most deployments use the defaults.



7. (Optional) Customize the text prompts used for username and password from the cf CLI and Apps Manager login popup by entering values for **Customize Username Label (on login page)** and **Customize Password Label (on login page)**.

8. (Optional) The **Proxy IPs Regular Expression** field contains a pipe-delimited set of regular expressions that UAA considers to be reverse proxy IP addresses. UAA respects the `x-forwarded-for` and `x-forwarded-proto` headers coming from IP addresses that match these regular expressions. To configure UAA to respond properly to Router or HAProxy requests coming from a public IP address, append a regular expression or regular expressions to match the public IP address.

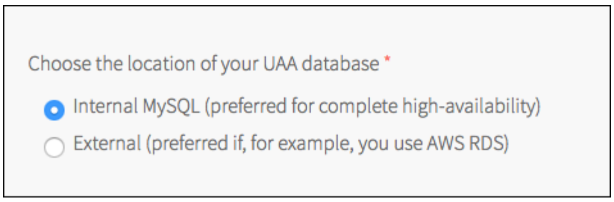
9. You can configure UAA to use the internal MySQL database provided with PCF, or you can configure an external database provider. Follow the procedures in either the Internal Database Configuration or the External Database Configuration section below.

> **Note**: For GCP installations, Pivotal recommends selecting **External** and using Google Cloud SQL.

> **Note**: If you are performing an upgrade, do not modify your existing internal database configuration or you may lose data. You must migrate your existing data before changing the configuration. See Upgrading Pivotal Cloud Foundry for additional upgrade information, and contact Pivotal Support for help.

Internal Database Configuration

1. Select **Internal MySQL**.

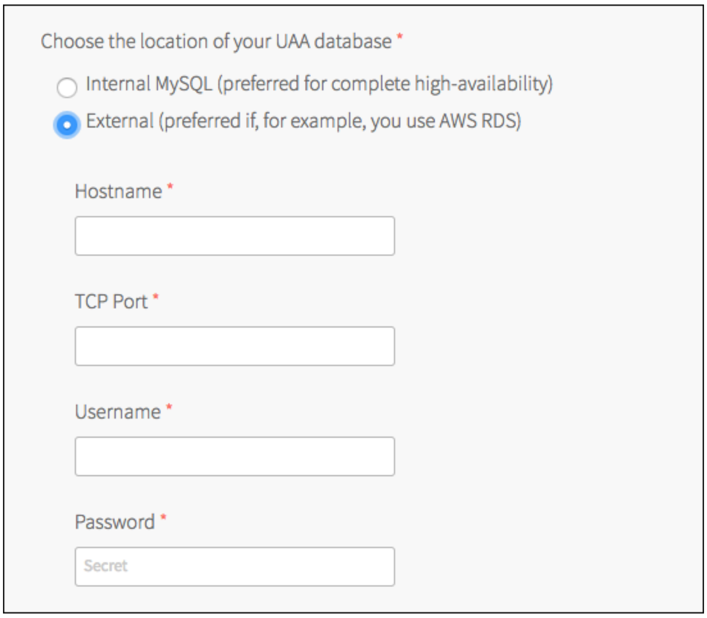


2. Click **Save**.

3. Ensure that you complete the “Configure Internal MySQL” step later in this topic to configure high availability and automatic backups for your internal MySQL databases.

External Database Configuration

1. From the **UAA** section in Pivotal Application Service (PAS), select **External**.



2. Enter the **Hostname** and **TCP Port** of the database from Step 6: Create Database Instance and Databases of the *Preparing to Deploy PCF on GCP* topic.

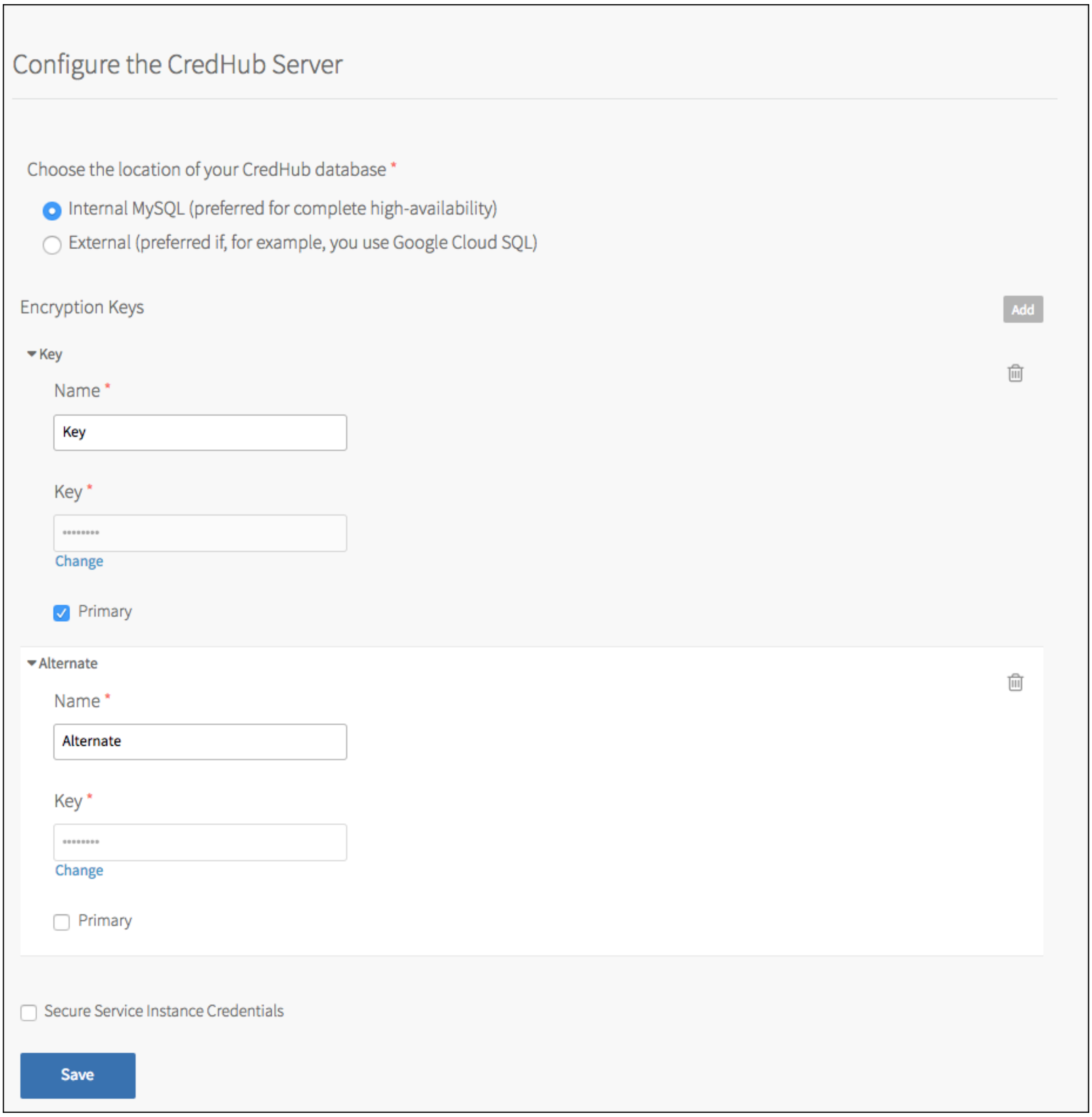
3. For **User Account and Authentication database username**, specify a unique username that can access this specific database on the database server.

4. For **User Account and Authentication database password**, specify a password for the provided username.

5. Click **Save**.

Step 11: (Optional) Configure CredHub

1. Select **Credhub**.



2. Choose the location of your CredHub Database. PAS includes this CredHub database for services to store their service instance credentials.

   a. If you chose **External**, enter the following:
      - **Hostname**: The IP address of the Google Cloud SQL instance that you created in Step 6: Create Database Instance and Databases of the *Preparing to Deploy PCF on GCP* topic. You can obtain this address from the Instances dashboard of the **SQL** configuration page in the GCP Console.
      - **TCP Port**: The port of your database server, `3306`.
      - **Username**: The username that can access this specific database on the database server. You created users in Preparing to Deploy PCF on GCP.
      - **Password**: The password for the provided username.
      - **Database CA Certificate**: Enter a certificate to use for encrypting traffic to and from the database.

3. Under **Encryption Keys**, specify a key to use for encrypting and decrypting the values stored in the CredHub database.
   - **Name**: Enter the name of the key.
   - **Key**: Enter a key that is at least 20 characters in length.
   - **Primary**: Select this checkbox to use this key as your primary key.

   > **Note**: Ensure that you only mark one key as **Primary**. The UI includes an **Add** button to add more keys to support key rotation. For more information, see the Rotating PAS CredHub Encryption Keys topic.

4. If your deployment uses any PCF services that support storing service instance credentials in CredHub and you want to enable this feature, select the **Secure Service Instance Credentials** checkbox.

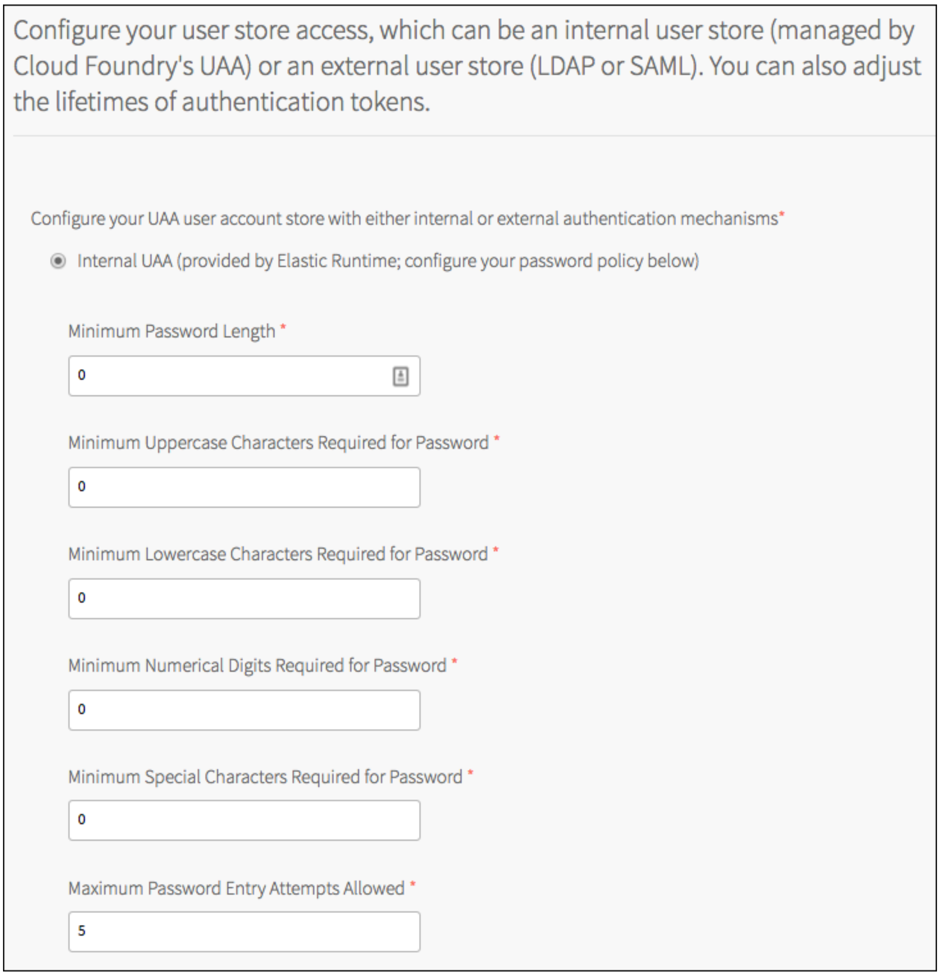
5. Select the **Resource Config** pane.

6. Under the **Job** column of the **CredHub** row, set the number of instances to `2`. This is the minimum instance count required for high availability.

> **Note**: To use the Runtime CredHub feature, you must follow the additional steps in Securing Service Instance Credentials with Runtime CredHub.

Step 12: Configure Authentication and Enterprise SSO

1. Select **Authentication and Enterprise SSO**.



2. To authenticate user sign-ons, your deployment can use one of three types of user database: the UAA server's internal user store, an external SAML identity provider, or an external LDAP server.
   - To use the internal UAA, select the **Internal** option and follow the instructions in the Configuring UAA Password Policy topic to configure your password policy.
   - To connect to an external identity provider through SAML, scroll down to select the **SAML Identity Provider** option and follow the instructions in the Configuring PCF for SAML section of the *Configuring Authentication and Enterprise SSO for Pivotal Application Service (PAS)* topic.
   - To connect to an external LDAP server, scroll down to select the **LDAP Server** option and follow the instructions in the Configuring LDAP section of the *Configuring Authentication and Enterprise SSO for PAS* topic.

3. Click **Save**.

Step 13: Configure System Databases

You can configure PAS to use Google Cloud SQL for the databases required by PAS.

> **Note**: If you are performing an upgrade, do not modify your existing internal database configuration or you may lose data. You must migrate your existing data first before changing the configuration. Contact Pivotal Support for help. See Upgrading Pivotal Cloud Foundry for additional

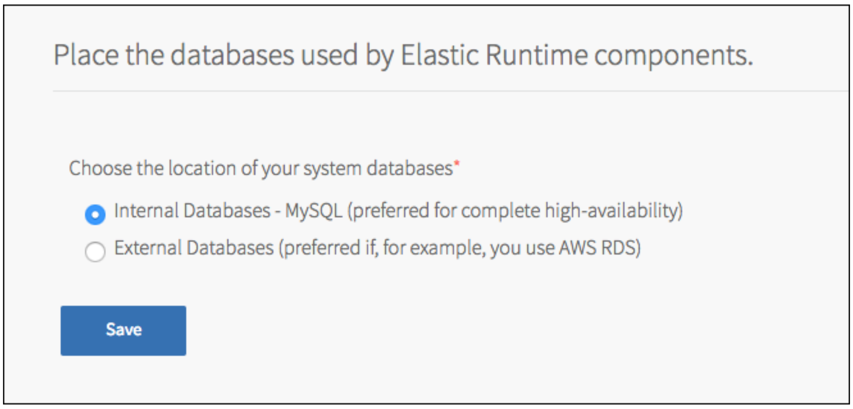
upgrade information.

Internal Database Configuration

Note: For GCP installations, Pivotal recommends selecting **External** and using Google Cloud SQL. Only use internal MySQL for non-production or test installations on GCP.

If you want to use internal databases for your deployment, perform the following steps:

1. Select **Databases**.



2. Select **Internal Databases - MySQL**.
3. Click **Save**.

Then proceed to Step 14: (Optional) Configure Internal MySQL to configure high availability and automatic backups for your internal MySQL databases.

External Database Configuration

Pivotal recommends using an external database such as Google Cloud SQL for high availability reasons.

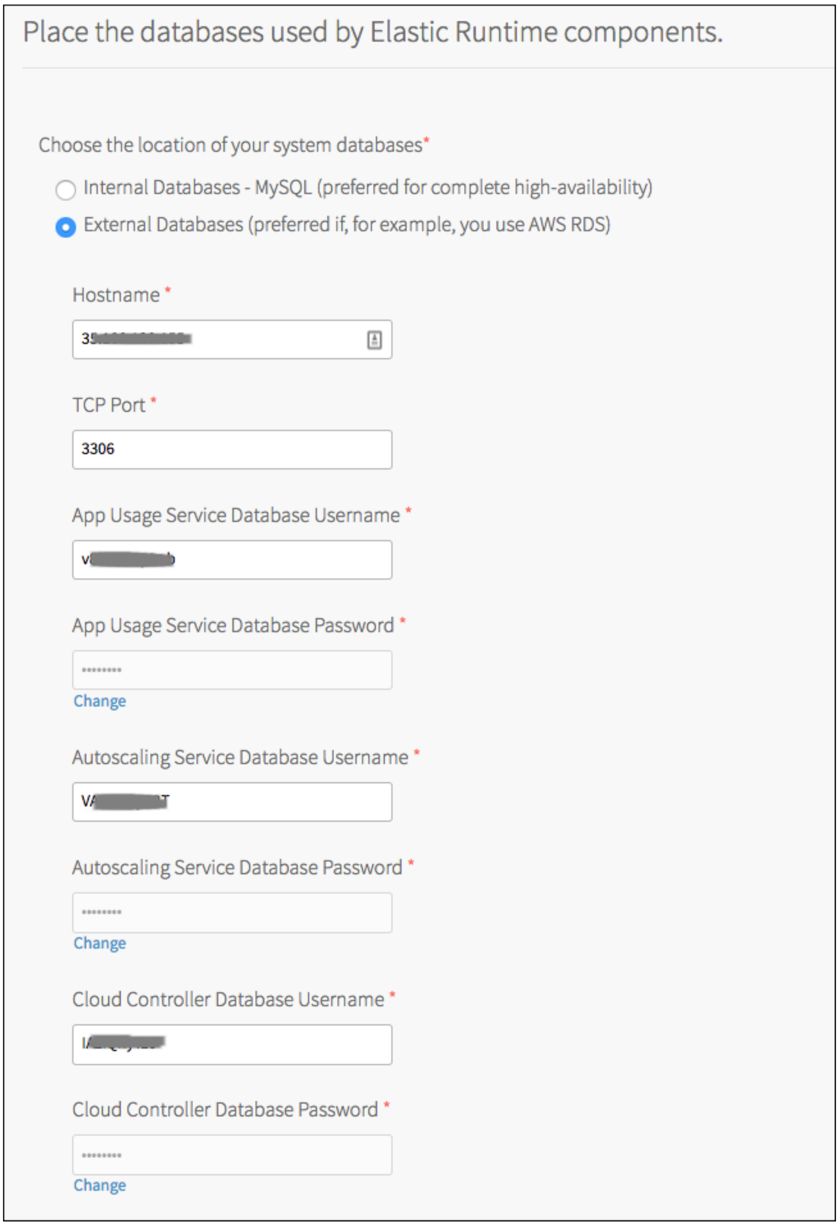
On GCP, you can use Google Cloud SQL and use the automated backup and high availability replica features.

Note: To configure an external database for UAA, see the *External Database Configuration* section of Configure UAA.

Warning: Protect whichever database you use in your deployment with a password.

To specify your PAS databases, perform the following steps:

1. Select the **External Databases** option.
2. In the **Hostname** field, enter the IP address of the Google Cloud SQL instance that you created in Step 6: Create Database Instance and Databases of the *Preparing to Deploy PCF on GCP* topic. You can obtain this address from the Instances dashboard of the **SQL** configuration page in the GCP Console.
3. In the **TCP Port** field, enter `3306`.
4. Each component that requires a relational database has two corresponding fields: one for the database username and one for the database password. For each set of fields, specify the username that can access this specific database on the database server and a password for the provided username. You created these users in Preparing to Deploy PCF on GCP.



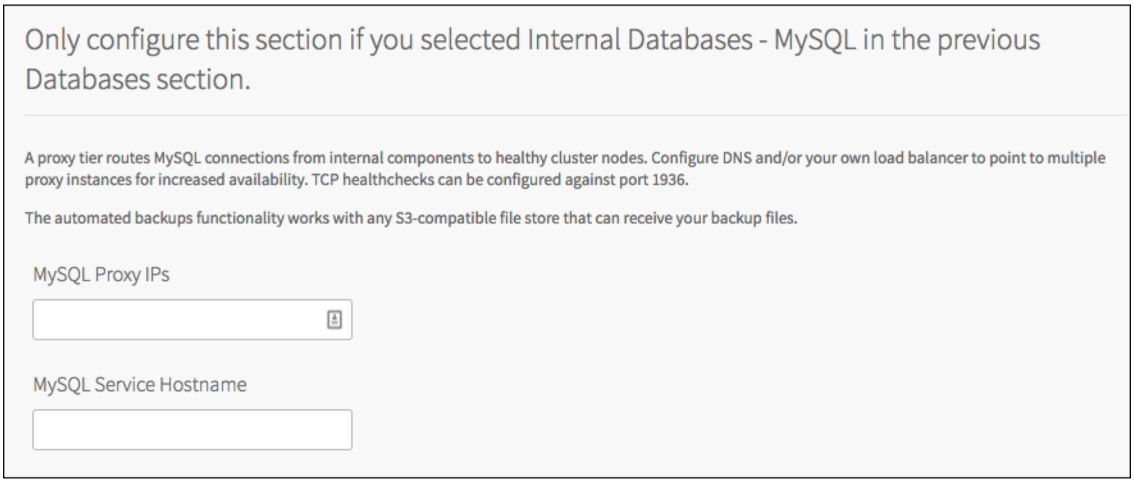
5. Click **Save**.

Step 14: (Optional) Configure Internal MySQL

Note: You only need to configure this section if you have selected **Internal Databases - MySQL** in the **Databases** section.

1. Select **Internal MySQL**.

2. In the **MySQL Proxy IPs** field, enter one or more comma-delimited IP addresses that are not in the reserved CIDR range of your network. If a MySQL node fails, these proxies re-route connections to a healthy node. See the MySQL Proxy topic for more information.



3. For **MySQL Service Hostname**, enter an IP address or hostname for your load balancer. If a MySQL proxy fails, the load balancer re-routes connections to a healthy proxy. If you leave this field blank, components are configured with the IP address of the first proxy instance entered above.

Warning: You must configure a load balancer to achieve complete high-availability.

1. In the **Replication canary time period** field, leave the default of 30 seconds or modify the value based on the needs of your deployment. Lower numbers cause the canary to run more frequently, which means that the canary reacts more quickly to replication failure but adds load to the database.

2. In the **Replication canary read delay** field, leave the default of 20 seconds or modify the value based on the needs of your deployment. This field configures how long the canary waits, in seconds, before verifying that data is replicating across each MySQL node. Clusters under heavy load can experience a small replication lag as write-sets are committed across the nodes.

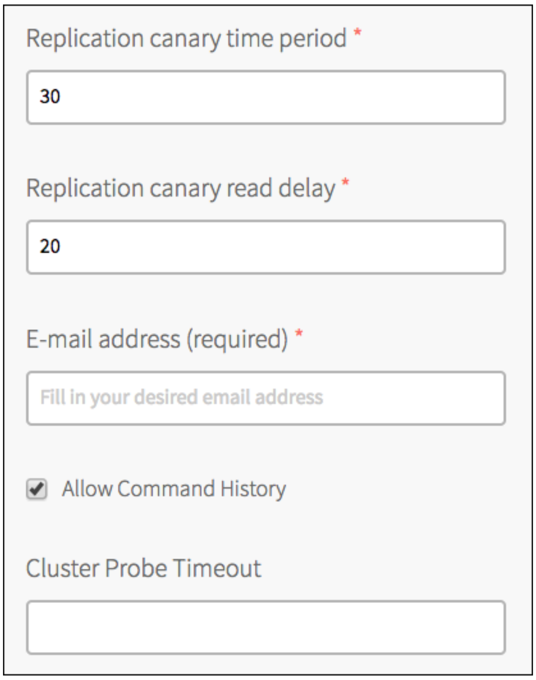
3. (**Required**): In the **E-mail address** field, enter the email address where the MySQL service sends alerts when the cluster experiences a replication issue or when a node is not allowed to auto-rejoin the cluster.

4. To prohibit the creation of command line history files on the MySQL nodes, deselect the **Allow Command History** checkbox.

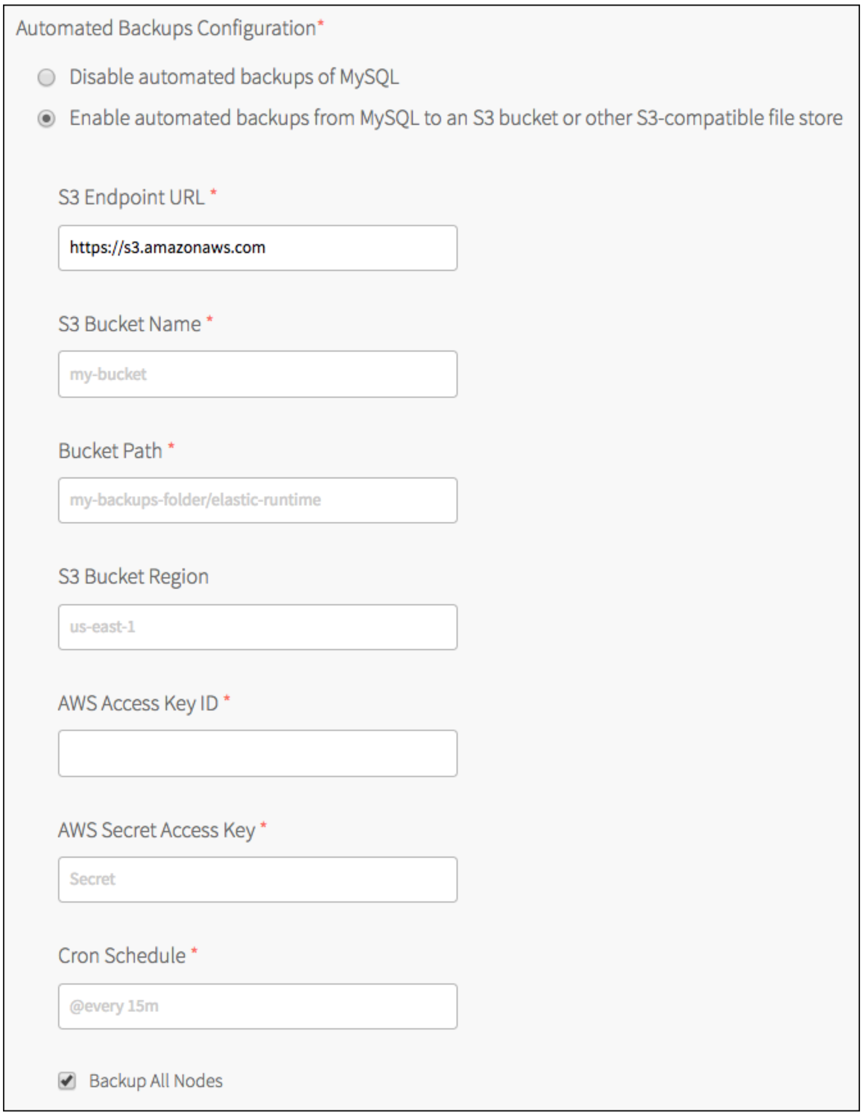
5. To allow the admin and roadmin to connect from any remote host, enable the **Allow Remote Admin Access** checkbox. When the checkbox is disabled, admins must `bosh ssh` into each MySQL VM to connect as the MySQL super user.

   Note: Network configuration and Application Security Groups restrictions may still limit a client's ability to establish a connection with the databases.

6. For **Cluster Probe Timeout**, enter the maximum amount of time, in seconds, that a new node will search for existing cluster nodes. If left blank, the default value is 10 seconds.

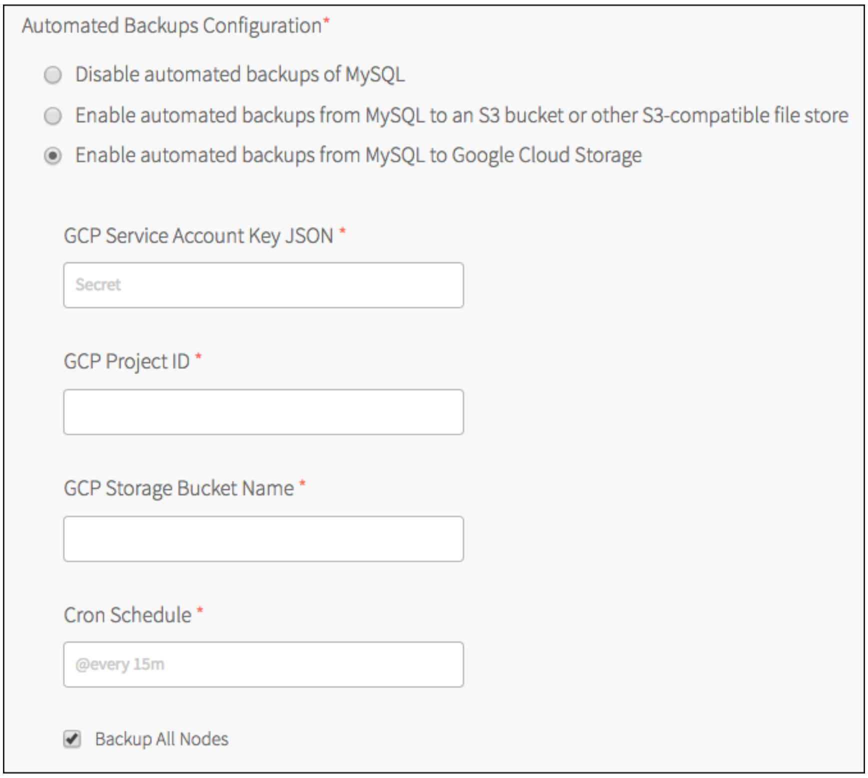


1. Under **Automated Backups Configuration**, choose one of five options for MySQL backups:
   - **Disable automatic backups of MySQL** disables automatic backups, but you can still deploy the Backup Prepare Node if you use BOSH Backup and Restore to back up your MySQL database. For more information, see the Backing Up Pivotal Cloud Foundry with BBR topic.
   - **Enable automated backups from MySQL to an S3 bucket or other S3-compatible file store** saves your backups to an existing Amazon Web Services (AWS) or Ceph S3-compatible blobstore.



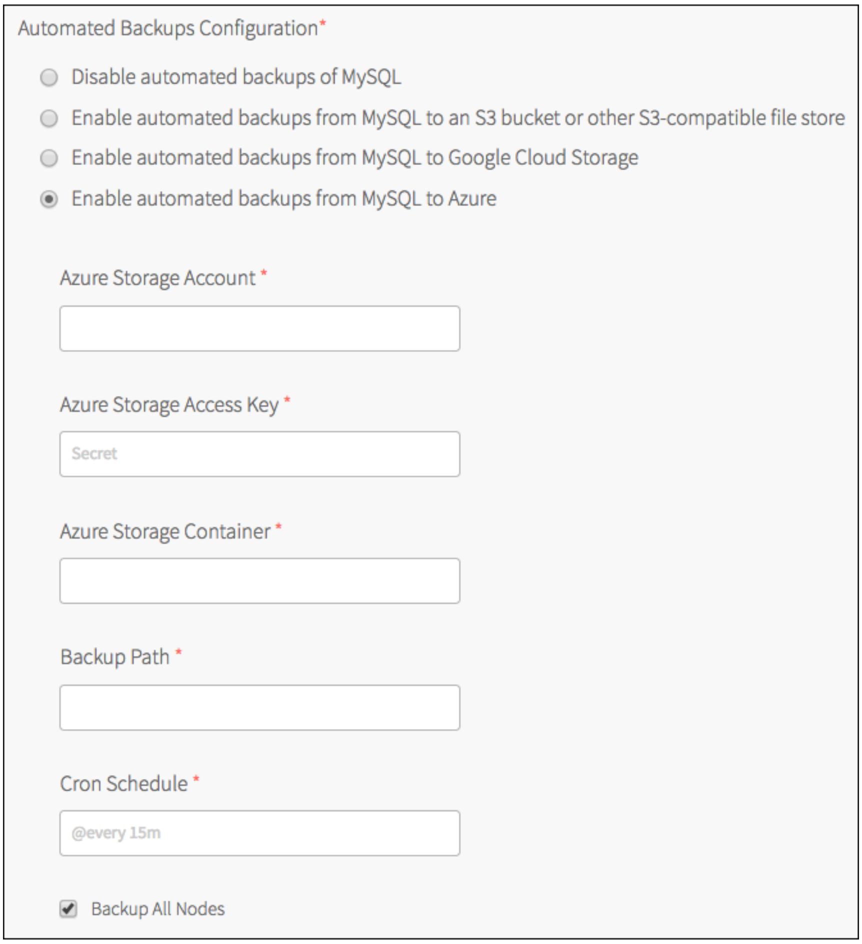
This option requires the following fields:

- For **S3 Bucket Name**, enter the name of your S3 bucket. Do not include an `s3://` prefix, a trailing `/`, or underscores. If the bucket does not already exist, it will be created automatically.
- For **Bucket Path**, specify a folder within the bucket to hold your MySQL backups. Do not include a trailing `/`.
- For **S3 Bucket Region**, enter the AWS region where the bucket is located, such as `us-east-1`.
- For **AWS Access Key ID** and **AWS Secret Access Key**, enter your AWS or Ceph credentials.
- For **Cron Schedule**, enter a valid cron expression to schedule your automated backups. Cron uses your computer's local time zone.
- Enable **Backup All Nodes** to make unique backups from each instance of the MySQL server rather than just the first MySQL server instance.
- **Enable automated backups from MySQL to Google Cloud Storage** saves your backups to Google Cloud Storage.



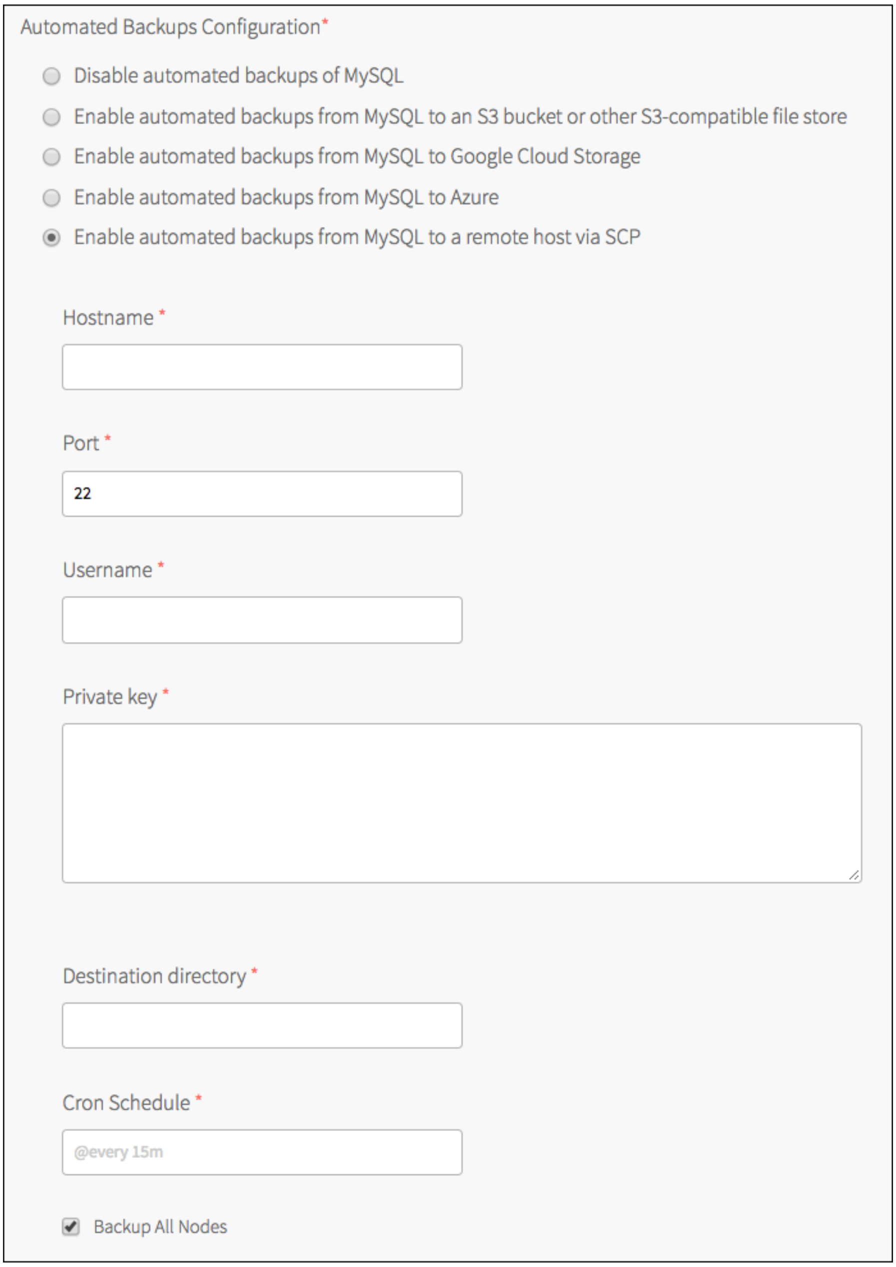
This option requires the following fields:

- For **GCP Service Account Key JSON**, enter the name of a Google Cloud Platform (GCP) service account key with access to the project and bucket specified below. This key must be in JSON format.
- For **GCP Project ID**, enter the project ID of your GCP project. You can find the project ID on the Dashboard of the GCP Console.
- For **GCP Storage Bucket Name**, enter the name of a bucket in Google Cloud Storage where your backups will be uploaded. If the bucket does not already exist, it will be created automatically.
- For **Cron Schedule**, enter a valid cron expression to schedule your automated backups. Cron uses your computer's local time zone.
- Enable **Backup All Nodes** to make unique backups from each instance of the MySQL server rather than just the first MySQL server instance.
- **Enable automated backups from MySQL to Azure** saves your backups to Azure.



This option requires the following fields:

- For **Azure Storage Account**, enter the name of an existing Azure storage account where backups will be uploaded. For more information about creating and managing an Azure storage account, see the Azure documentation.
- For **Azure Storage Access Key**, enter an Azure storage access key for the storage account.
- For **Azure Storage Container**, enter the name of an existing Azure storage container that will store the backups.
- For **Backup Path**, enter the path within the Azure storage container where backups will be uploaded.
- For **Cron Schedule**, enter a valid cron expression to schedule your automated backups. Cron uses your computer's local time zone.
- Enable **Backup All Nodes** to make unique backups from each instance of the MySQL server rather than just the first MySQL server instance.
- **Enable automated backups from MySQL to a remote host via SCP**saves your backups to a remote host using secure copy protocol (SCP).



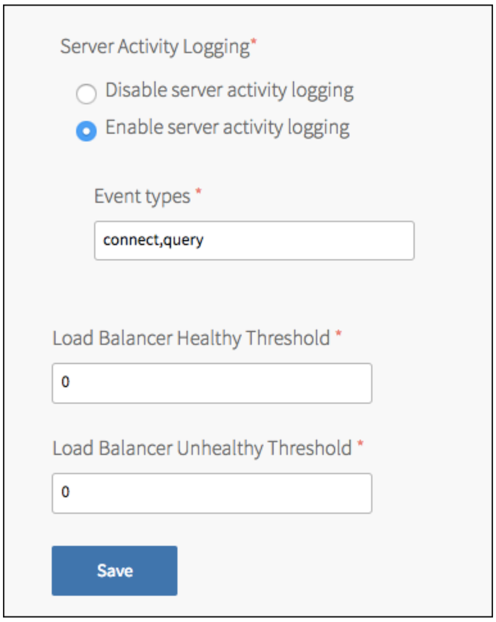
This option requires the following fields:

- For **Hostname**, enter the name of your SCP host.
- For **Port**, enter your SCP port. This should be the TCP port that your SCP host uses for SSH. The default port is `22`.
- For **Username**, enter your SSH username for the SCP host.
- For **Private key**, paste in your SSH private key.
- For **Destination directory**, enter the directory on the SCP host where you want to save backup files.
- For **Cron Schedule**, enter a valid cron expression to schedule your automated backups. Cron uses your computer's local time zone.
- Enable **Backup All Nodes** to make unique backups from each instance of the MySQL server rather than just the first MySQL server instance.

> **Note**: If you choose to enable automated MySQL backups, set the number of instances for the **Backup Prepare Node** under the **Resource Config** section of the Pivotal Application Service (PAS) tile to `1`.

2. If you want to log audit events for internal MySQL, select **Enable server activity logging** under **Server Activity Logging**.

   a. For the **Event types** field, you can enter the events you want the MySQL service to log. By default, this field includes `connect` and `query`, which tracks who connects to the system and what queries are processed. For more information, see the Logging Events section of the MariaDB documentation.



3. Enter values for the following fields:
    - **Load Balancer Healthy Threshold**: Specifies the amount of time, in seconds, to wait until declaring the MySQL proxy instance started. This allows an external load balancer time to register the instance as healthy.
    - **Load Balancer Unhealthy Threshold**: Specifies the amount of time, in seconds, that the MySQL proxy continues to accept connections before shutting down. During this period, the healthcheck reports as unhealthy to cause load balancers to fail over to other proxies. You must enter a value greater than or equal to the maximum time it takes your load balancer to consider a proxy instance unhealthy, given repeated failed healthchecks.

4. If you want to enable the MySQL interruptor feature, select the checkbox to **Prevent node auto re-join**. This feature stops all writes to the MySQL database if it notices an inconsistency in the dataset between the nodes. For more information, see the Interruptor section in the MySQL for PCF documentation.

5. Click **Save**.

Step 15: Configure File Storage

To minimize system downtime, Pivotal recommends using highly resilient and redundant*external* filestores for your Pivotal Application Service (PAS) file storage.

When configuring file storage for the Cloud Controller in PAS, you can select one of the following:

- Internal WebDAV filestore
- External S3-compatible or Ceph-compatible filestore
- External Google Cloud Storage
- External Azure Cloud Storage

For production-level PCF deployments on GCP, Pivotal recommends selecting **External Google Cloud Storage**. For more information about production-level PCF deployments on GCP, see the Reference Architecture for Pivotal Cloud Foundry on GCP.

For additional factors to consider when selecting file storage, see the Considerations for Selecting File Storage in Pivotal Cloud Foundry topic.

Internal Filestore

Internal file storage is only appropriate for small, non-production deployments.

To use the PCF internal filestore, perform the following steps:

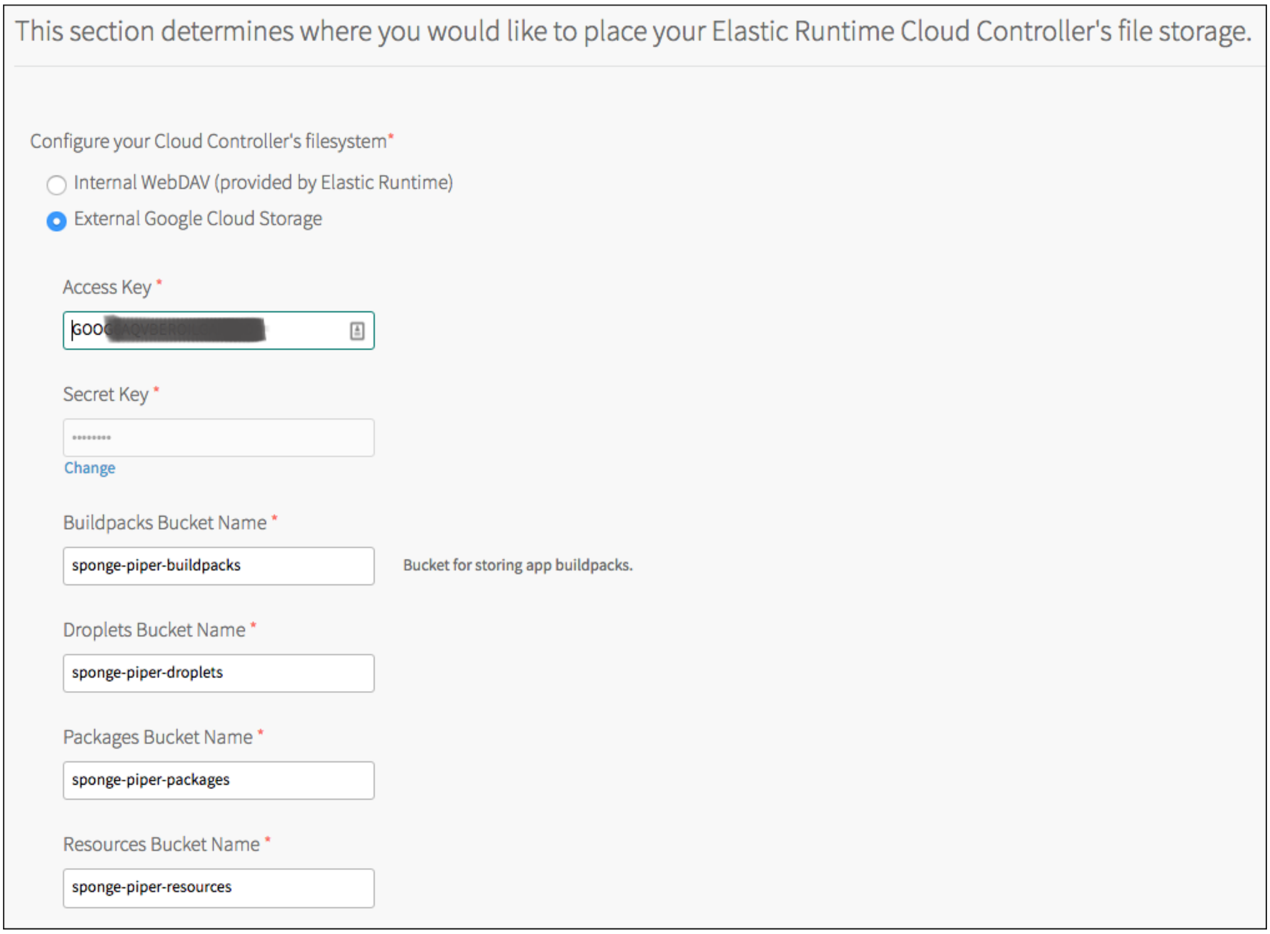
1. In the Pivotal Application Service (PAS) tile, select **File Storage**.

2. Select **Internal WebDAV**, and click **Save**.

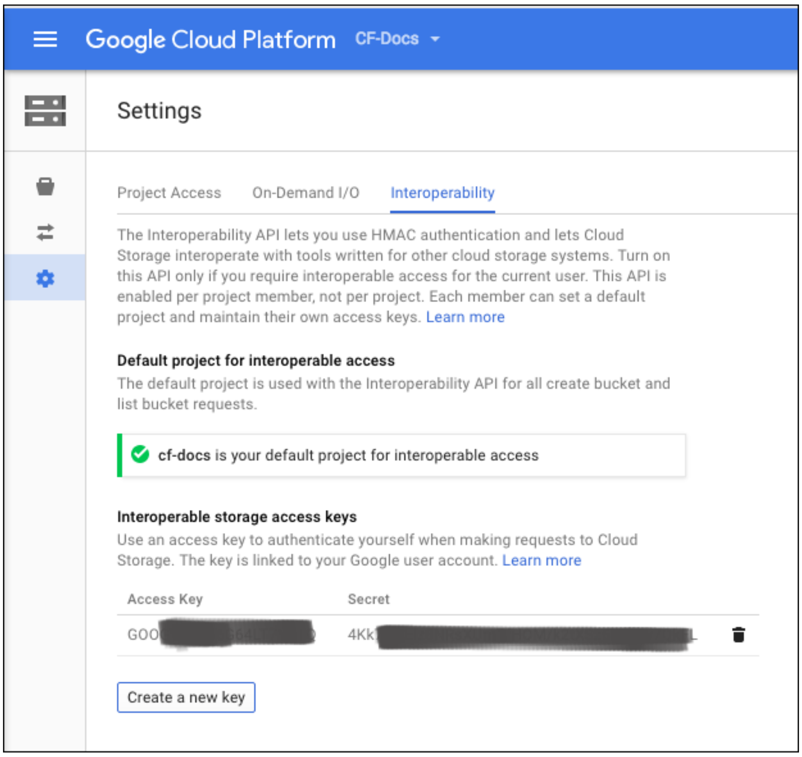
External Google Cloud Storage

To use external Google file storage for your Pivotal Application Service (PAS) filestore, perform the following steps:

1. Select the **External Google Cloud Storage** option.



2. Enter values for **Access Key** and **Secret Key**. To obtain the values for these fields:
   - In the GCP Console, navigate to the **Storage** tab, then click **Settings**.
   - Click **Interoperability**.
   - If necessary, click **Enable interoperability access**. If interoperability access is already enabled, confirm that the default project matches the project where you are installing PCF.



- Click **Create a new key.**
- Copy and paste the generated values into the corresponding PAS fields. PCF uses these values for authentication when connecting to Google Cloud Storage.

3. Enter the names of the storage buckets you created in [Preparing to Deploy PCF on GCP](#):
   - **Buildpacks Bucket Name:** `MY-PCF-buildpacks`
   - **Droplets Bucket Name:** `MY-PCF-droplets`
   - **Resources Bucket Name:** `MY-PCF-resources`
   - **Packages Bucket Name:** `MY-PCF-packages`

4. Click **Save.**

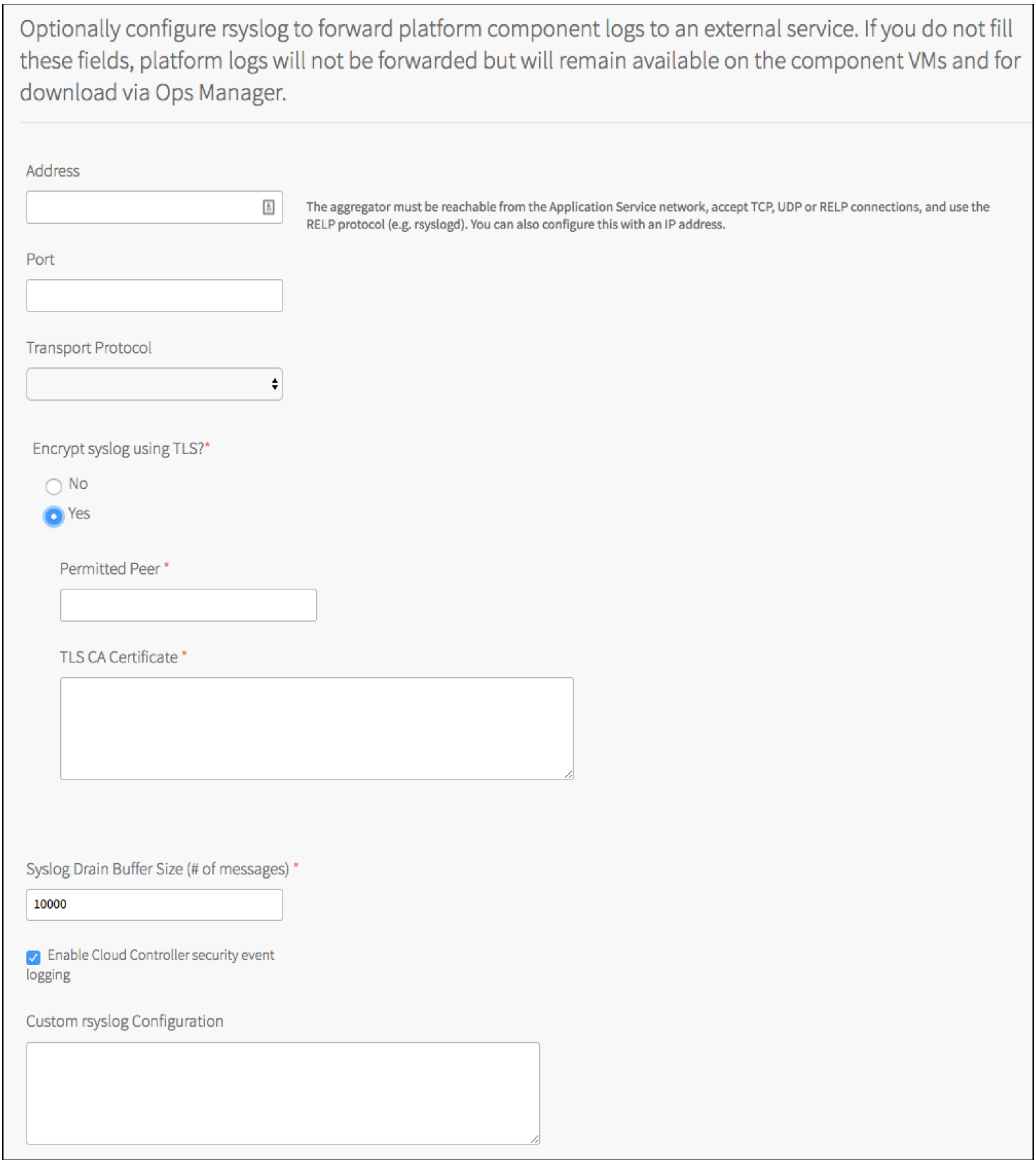
Other IaaS Storage Options

Azure Storage and External S3-Compatible File Storage are also available as file storage options, but Pivotal does not recommend these for a typical PCF on GCP installation.

Step 16: (Optional) Configure System Logging

If you forward logging messages to an external Reliable Event Logging Protocol (RELP) server, complete the following steps:

1. Select the **System Logging** section that is located within your PAS **Settings** tab.



2. Enter the IP address of your syslog server in **Address**.

3. Enter the port of your syslog server in **Port**. The default port for a syslog server is `514`.

> **Note**: The host must be reachable from the PAS network, accept TCP connections, and use the RELP protocol. Ensure your syslog server listens on external interfaces.

4. Select a **Transport Protocol** to use when forwarding logs.

5. If you plan to use TLS encryption when sending logs to the remote server, select **Yes** when answering the **Encrypt syslog using TLS?** question.

    a. In the **Permitted Peer** field, enter either the name or SHA1 fingerprint of the remote peer.
    b. In the **TLS CA Certificate** field, enter the TLS CA Certificate for the remote server.

6. For the **Syslog Drain Buffer Size**, enter the number of messages the Doppler server can hold from Metron agents before the server starts to drop them. See the Loggregator Guide for Cloud Foundry Operators topic for more details.

7. If you want to include security events in your log stream, select the **Enable Cloud Controller security event logging** checkbox. This logs all API requests, including the endpoint, user, source IP address, and request result, in the Common Event Format (CEF).

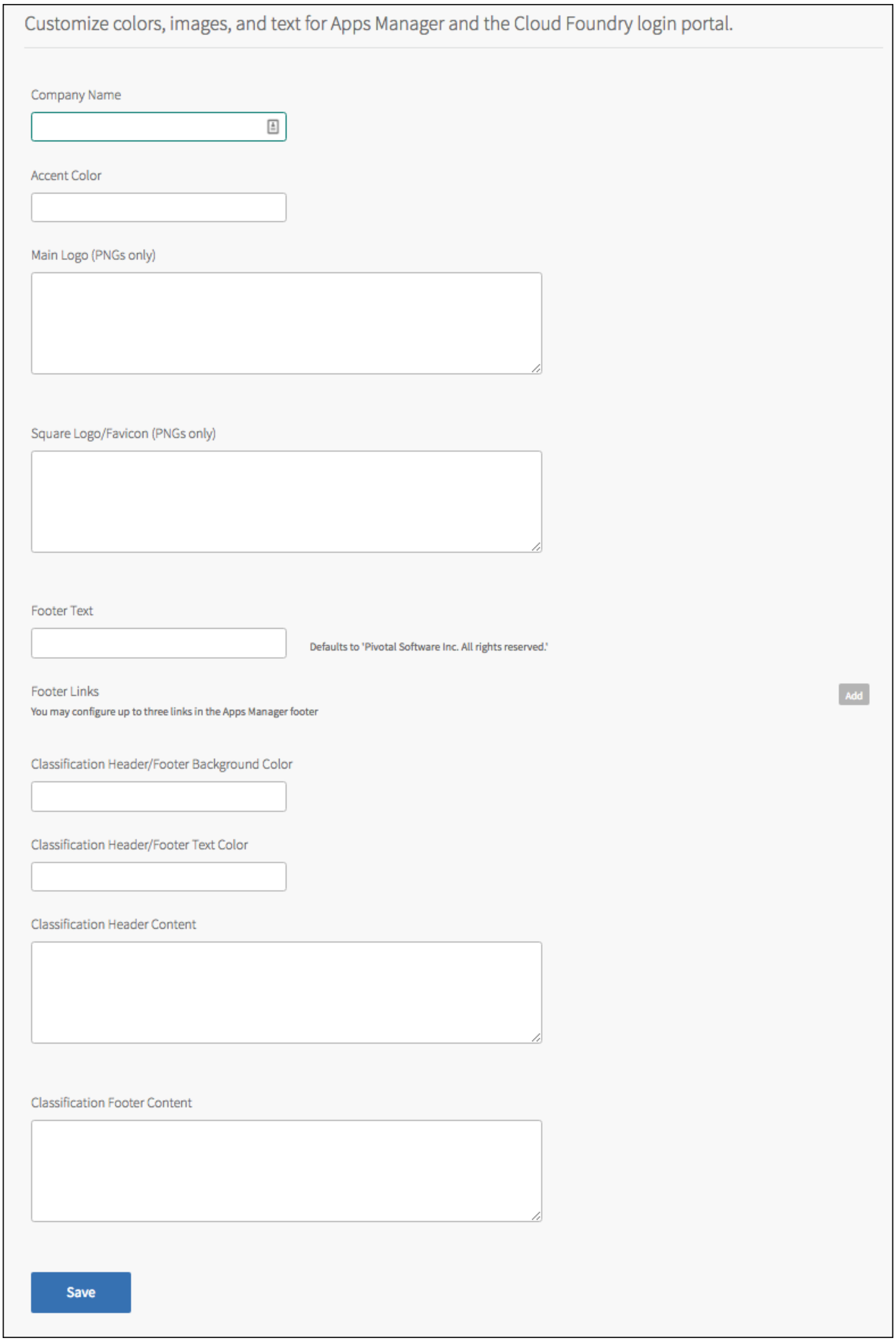
8. If you want to specify a custom syslog formatting rule, enter it in the **Custom rsyslog Configuration** field in RainerScript syntax.

9. Click **Save**.

Step 17: (Optional) Customize Apps Manager

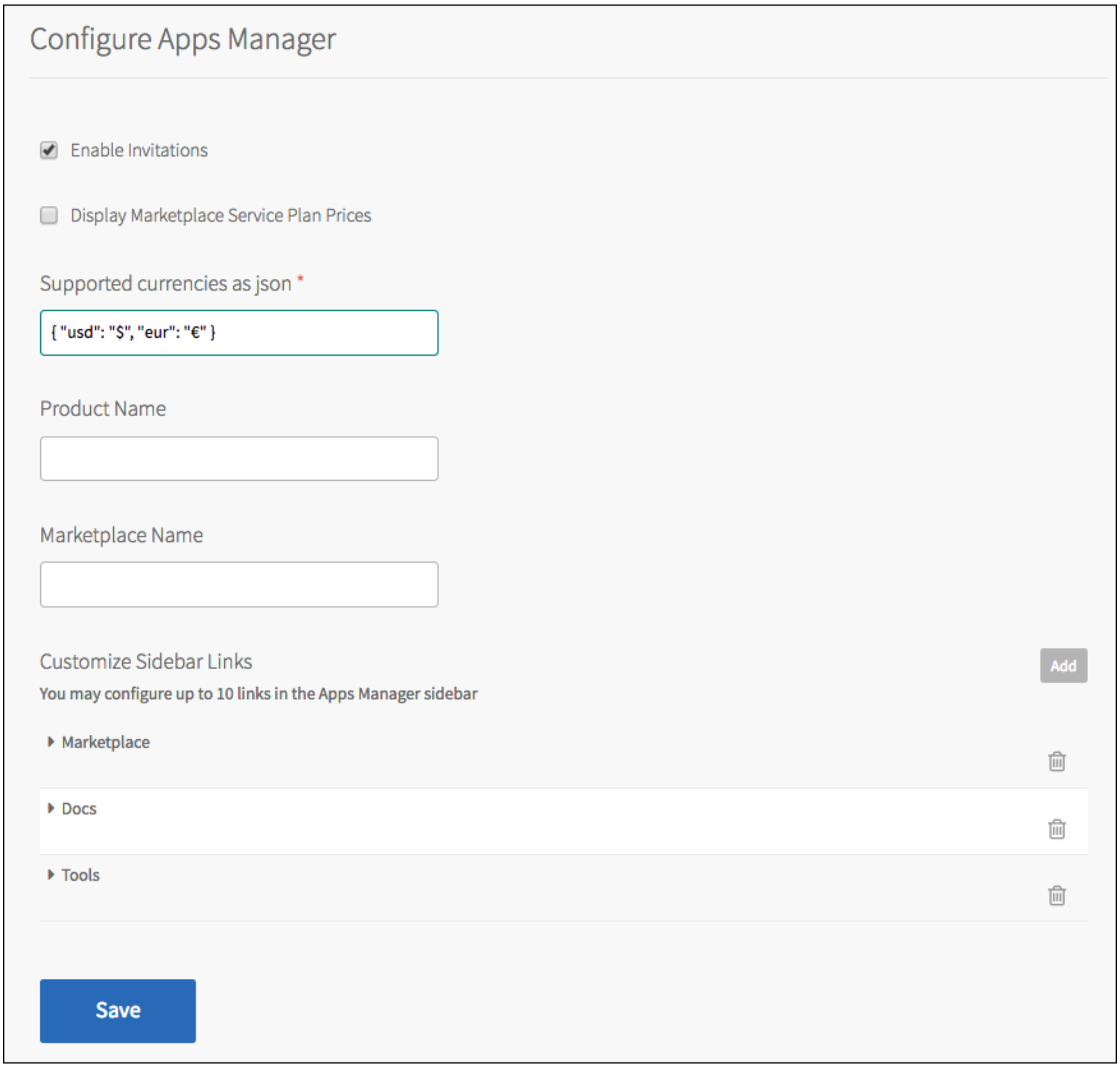
The **Custom Branding** and **Apps Manager** sections customize the appearance and functionality of Apps Manager. Refer to Custom Branding Apps Manager for descriptions of the fields on these pages and for more information about customizing Apps Manager.

1. Select **Custom Branding**. Use this section to configure the text, colors, and images of the interface that developers see when they log in, create an account, reset their password, or use Apps Manager.



2. Click **Save** to save your settings in this section.

3. Select **Apps Manager**.



4. Select **Enable Invitations** to enable invitations in Apps Manager. Space Managers can invite new users for a given space, Org Managers can invite new users for a given org, and Admins can invite new users across all orgs and spaces. See the Inviting New Users section of the *Managing User Roles with Apps Manager* topic for more information.

5. Select **Display Marketplace Service Plan Prices** to display the prices for your services plans in the Marketplace.

6. Enter the **Supported currencies as json** to appear in the Marketplace. Use the format `{"CURRENCY-CODE":"SYMBOL"}`. This defaults to `{"usd": "$", "eur": "€"}`.

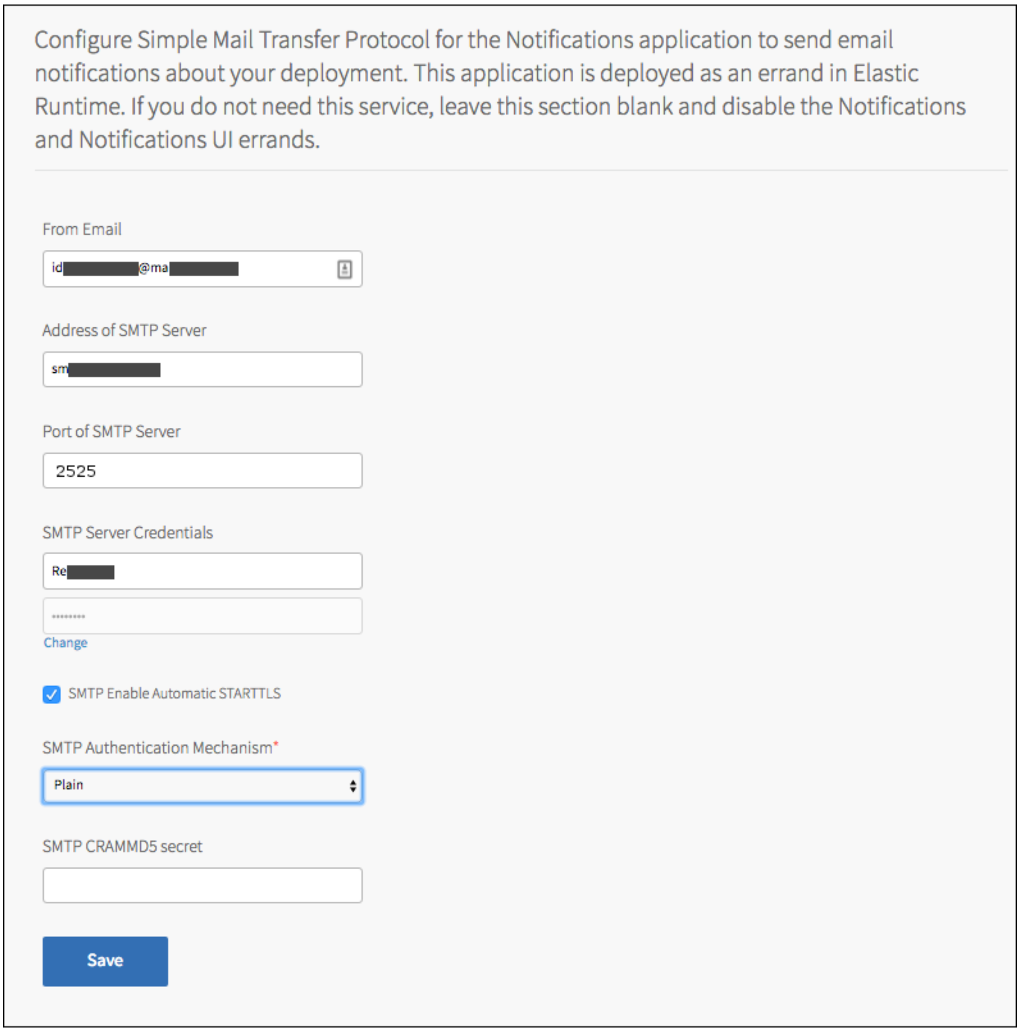
7. Use **Product Name**, **Marketplace Name**, and **Customize Sidebar Links** to configure page names and sidebar links in the **Apps Manager** and **Marketplace** pages.

8. Click **Save** to save your settings in this section.

Step 18: (Optional) Configure Email Notifications

PAS uses SMTP to send invitations and confirmations to Apps Manager users. You must complete the **Email Notifications** page if you want to enable end-user self-registration.

1. Select **Email Notifications**.



2. Enter your reply-to and SMTP email information. You must use port `2525`. Ports `25` and `587` are not allowed on GCP Compute Engine.

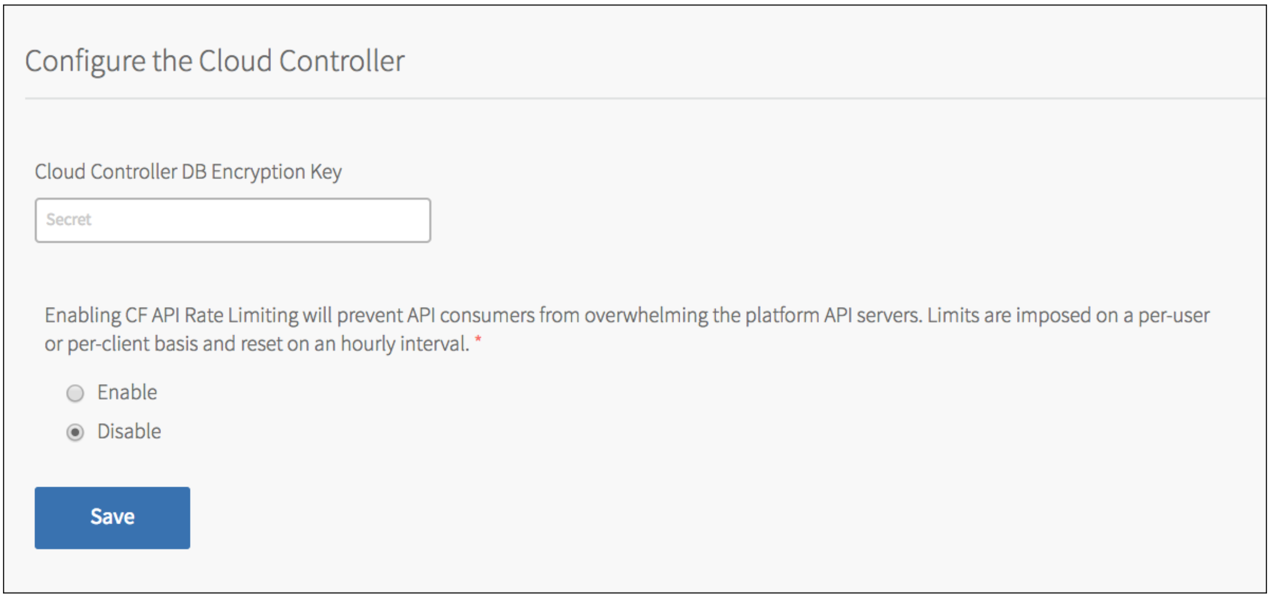
3. For **SMTP Authentication Mechanism**, select `none`.

4. Click **Save**.

> **Note**: If you do not configure the SMTP settings using this form, the administrator must create orgs and users using the cf CLI tool. See Creating and Managing Users with the cf CLI for more information.

Step 19: Configure Cloud Controller

1. Click **Cloud Controller**.



2. Enter your **Cloud Controller DB Encryption Key** if all of the following are true:
   - You deployed Pivotal Application Service (PAS) previously.
   - You then stopped PAS or it crashed.
   - You are re-deploying PAS with a backup of your Cloud Controller database.

   See [Backing Up Pivotal Cloud Foundry]() for more information.

3. CF API Rate Limiting prevents API consumers from overwhelming the platform API servers. Limits are imposed on a per-user or per-client basis and reset on an hourly interval.

   To disable CF API Rate Limiting, select **Disable** under **Enable CF API Rate Limiting**. To enable CF API Rate Limiting, perform the following steps:

   a. Under **Enable CF API Rate Limiting**, select **Enable**.
   b. For **General Limit**, enter the number of requests a user or client is allowed to make over an hour interval for all endpoints that do not have a custom limit. The default value is `2000`.
   c. For **Unauthenticated Limit**, enter the number of requests an unauthenticated client is allowed to make over an hour interval. The default value is `100`.

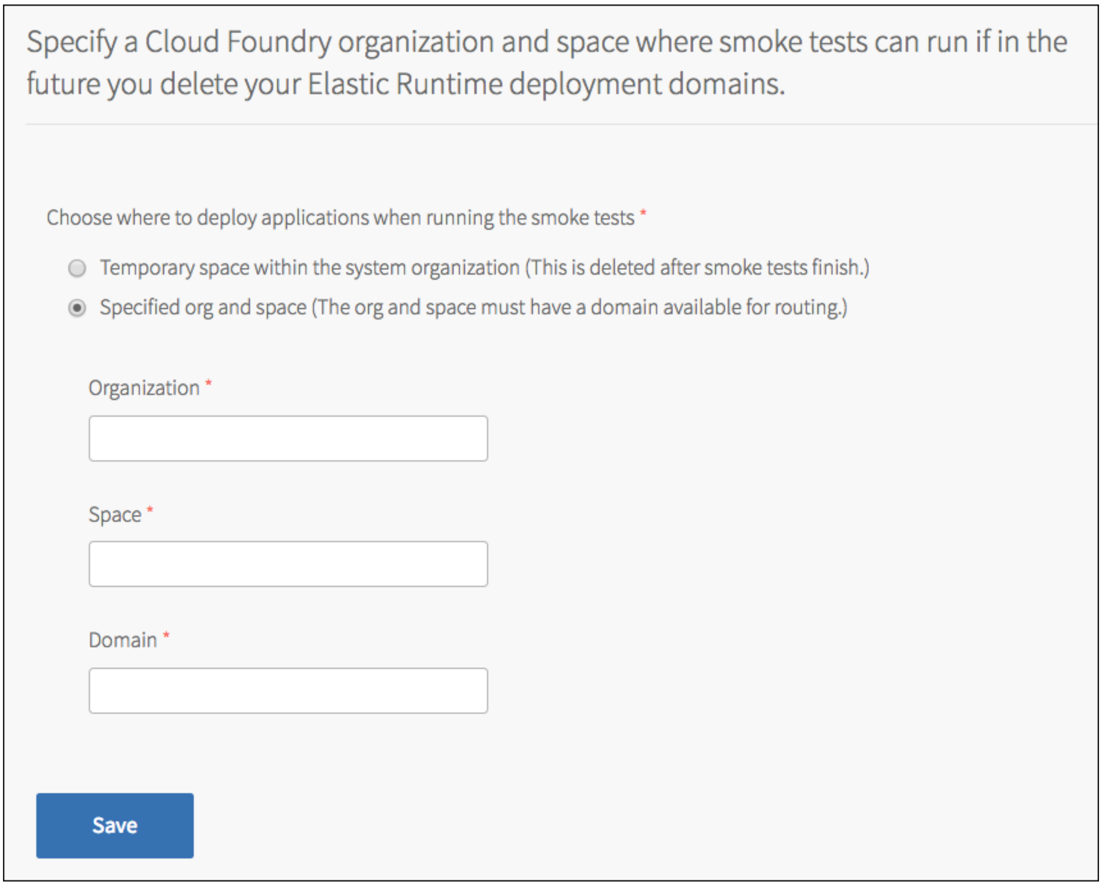
4. Click **Save**.

Step 20: Configure Smoke Tests

The Smoke Tests errand runs basic functionality tests against your Pivotal Application Service (PAS) deployment after an installation or update. In this section, choose where to run smoke tests. In the **Errands** section, you can choose whether or not to run the Smoke Tests errand.

1. Select **Smoke Tests**.

2. If you have a shared apps domain, select **Temporary space within the system organization**, which creates a temporary space within the `system` organization for running smoke tests and deletes the space afterwards. Otherwise, select **Specified org and space** and complete the fields to specify where you want to run smoke tests.



3. Click **Save**.

Step 21: (Optional) Enable Advanced Features

The **Advanced Features** section of Pivotal Application Service (PAS) includes new functionality that may have certain constraints. Although these features are fully supported, Pivotal recommends caution when using them in production environments.

Diego Cell Memory and Disk Overcommit

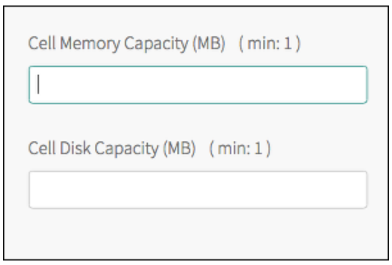
If your apps do not use the full allocation of disk space and memory set in the **Resource Config** tab, you might want use this feature. These fields control the amount to overcommit disk and memory resources to each Diego Cell VM.

For example, you might want to use the overcommit if your apps use a small amount of disk and memory capacity compared to the amounts set in the **Resource Config** settings for **Diego Cell**.

> **Note**: Due to the risk of app failure and the deployment-specific nature of disk and memory use, Pivotal has no recommendation about how much, if any, memory or disk space to overcommit.

To enable overcommit, follow these steps:

1. Select **Advanced Features**.



2. Enter the total desired amount of Diego cell memory value in the **Cell Memory Capacity (MB)** field. Refer to the **Diego Cell** row in the **Resource Config** tab for the current Cell memory capacity settings that this field overrides.

3. Enter the total desired amount of Diego cell disk capacity value in the **Cell Disk Capacity (MB)** field. Refer to the **Diego Cell** row in the **Resource Config** tab for the current Cell disk capacity settings that this field overrides.

4. Click **Save**.

> **Note**: Entries made to each of these two fields set the total amount of resources allocated, not the overage.

Whitelist for Non-RFC-1918 Private Networks

Some private networks require extra configuration so that internal file storage (WebDAV) can communicate with other PCF processes.

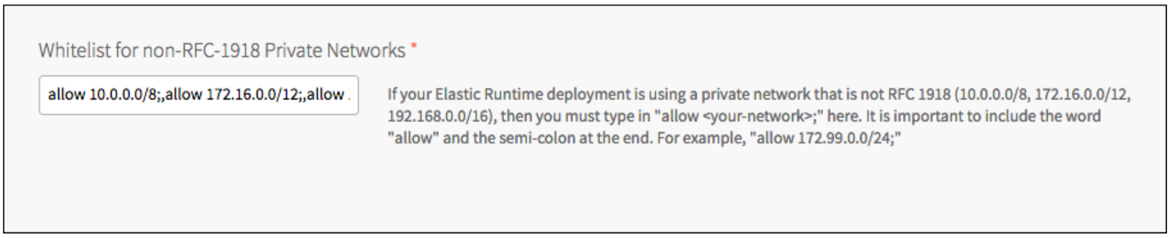
The **Whitelist for non-RFC-1918 Private Networks** field is provided for deployments that use a non-RFC 1918 private network. This is typically a private network other than `10.0.0.0/8`, `172.16.0.0/12`, or `192.168.0.0/16`.

Most PCF deployments do not require any modifications to this field.

To add your private network to the whitelist, perform the following steps:

1. Select **Advanced Features**.

2. Append a new `allow` rule to the existing contents of the **Whitelist for non-RFC-1918 Private Networks** field.



Include the word `allow`, the network CIDR range to allow, and a semi-colon (`;`) at the end. For example: `allow 172.99.0.0/24;`

3. Click **Save**.

CF CLI Connection Timeout

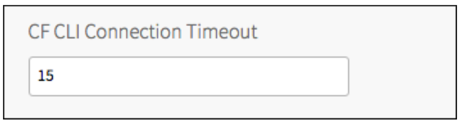
The **CF CLI Connection Timeout** field allows you to override the default five second timeout of the Cloud Foundry Command Line Interface (cf CLI) used within your PCF deployment. This timeout affects the cf CLI command used to push PAS errand apps such as Notifications, Autoscaler, and Apps Manager.

Set the value of this field to a higher value, in seconds, if you are experiencing domain name resolution timeouts when pushing errands in PAS.

To modify the value of the **CF CLI Connection Timeout**, perform the following steps:

1. Select **Advanced Features**.

2. Add a value, in seconds, to the **CF CLI Connection Timeout** field.



3. Click **Save**.

Step 22: Configure Errands

Errands are scripts that Ops Manager runs automatically when it installs or uninstalls a product, such as a new version of Pivotal Application Service (PAS). There are two types of errands: *post-deploy errands* run after the product is installed, and *pre-delete errands* run before the product in uninstalled.

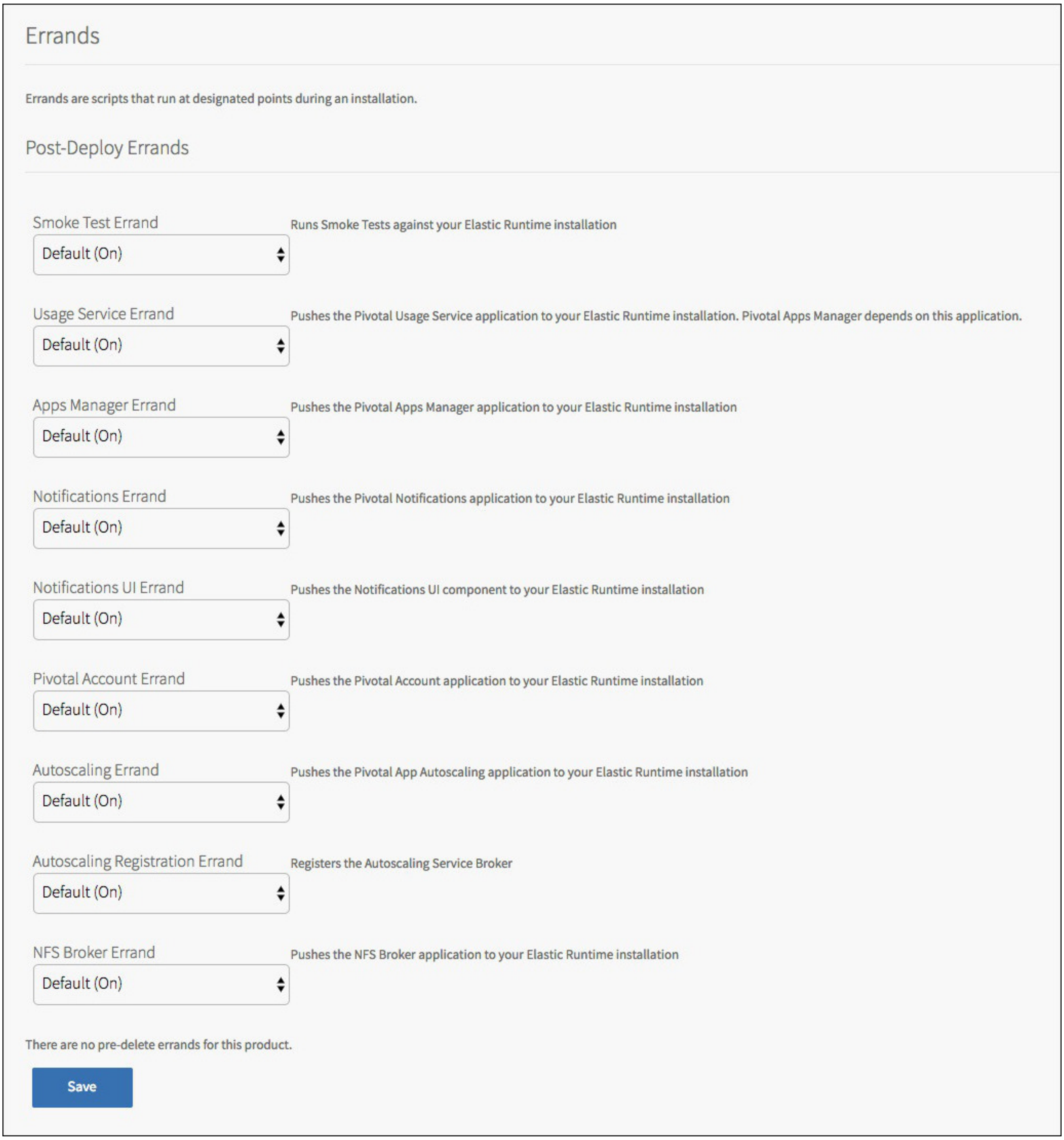
By default, Ops Manager always runs pre-delete errands, and only runs post-deploy errands when the product has changed since the last time Ops

Manager installed something. In PAS, the Smoke Test Errand defaults to always run.

The PAS tile **Errands** pane lets you change these run rules. For each errand, you can select **On** to run it always, **Off** to never run it, or **When Changed** to run it only when the product has changed since the last install.

For more information about how Ops Manager manages errands, see the [Managing Errands in Ops Manager] topic.

> **Note**: Several errands deploy apps that provide services for your deployment, such as Autoscaling and Notifications. Once one of these apps is running, selecting **Off** for the corresponding errand on a subsequent installation does not stop the app.



- **Smoke Test Errand** verifies that your deployment can do the following:
  - Push, scale, and delete apps
  - Create and delete orgs and spaces
- **Usage Service Errand** deploys the Pivotal Usage Service application, which Apps Manager depends on.
- **Apps Manager Errand** deploys Apps Manager, a dashboard for managing apps, services, orgs, users, and spaces. Until you deploy Apps Manager, you

must perform these functions through the cf CLI. After Apps Manager has been deployed, Pivotal recommends deselecting the checkbox for this errand on subsequent PAS deployments. For more information about Apps Manager, see the Getting Started with the Apps Manager topic.

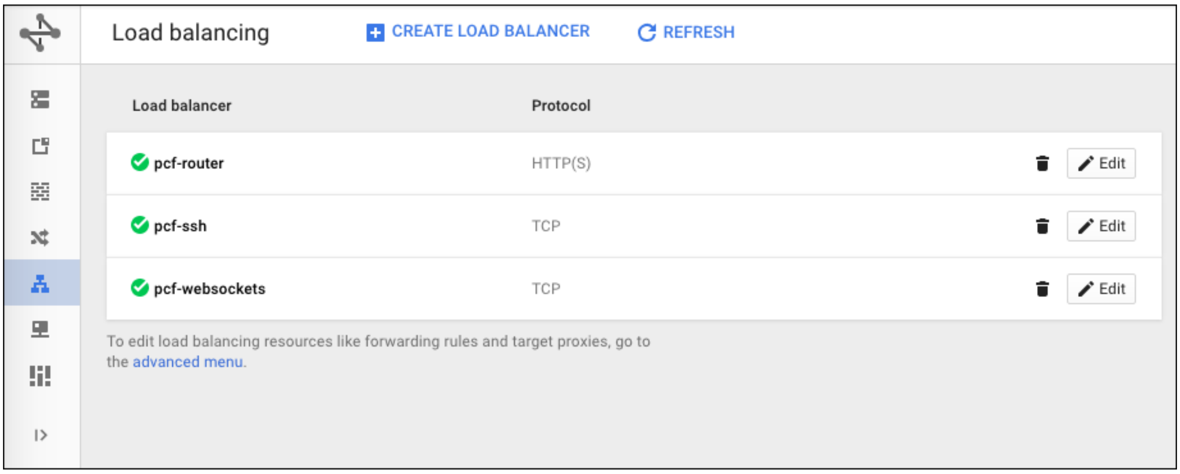
- **Notifications Errand** deploys an API for sending email notifications to your PCF platform users.

> **Note**: The Notifications app requires that you configure SMTP with a username and password, even if you set the value of **SMTP Authentication Mechanism** to `none`.

- **Notifications UI Errand** deploys a dashboard for users to manage notification subscriptions.
- **Pivotal Account Errand** deploys Pivotal Account, a dashboard that allows users to create and manage their accounts. In the Pivotal Account dashboard, users can launch applications, manage their profiles, manage account security, manage notifications, and manage approvals. See the Enabling Pivotal Account topic for more information.
- **Autoscaling Errand** enables you to configure your apps to automatically scale in response to changes in their usage load. See the Scaling an Application Using Autoscaler topic for more information.
- **Autoscaling Registration Errand** makes the Autoscaling service available to your applications. Without this errand, you cannot bind the Autoscaling app to your apps.
- **NFS Broker Errand** enables you to use NFS Volume Services by installing the NFS Broker app in PAS. See the Enabling NFS Volume Services topic for more information.

Step 23: Configure Load Balancers

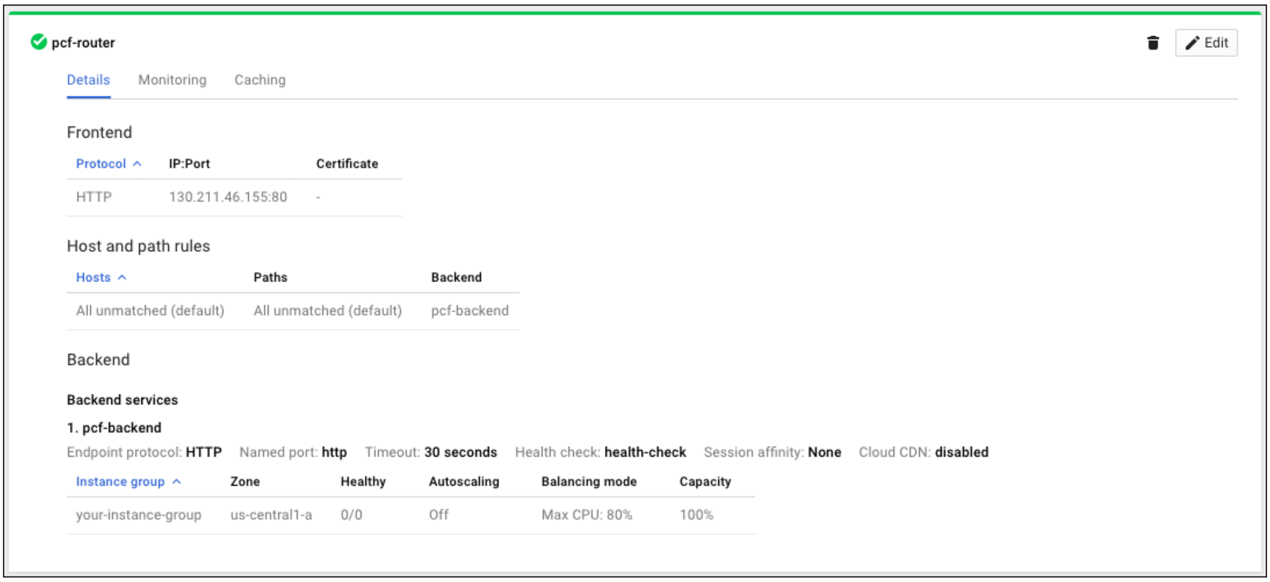
1. Navigate to the GCP Console and click **Load balancing**.



You should see the SSH load balancer, the HTTP(S) load balancer, the TCP WebSockets load balancer, and the TCP router that you created in the *Preparing to Deploy PCF on GCP* topic.

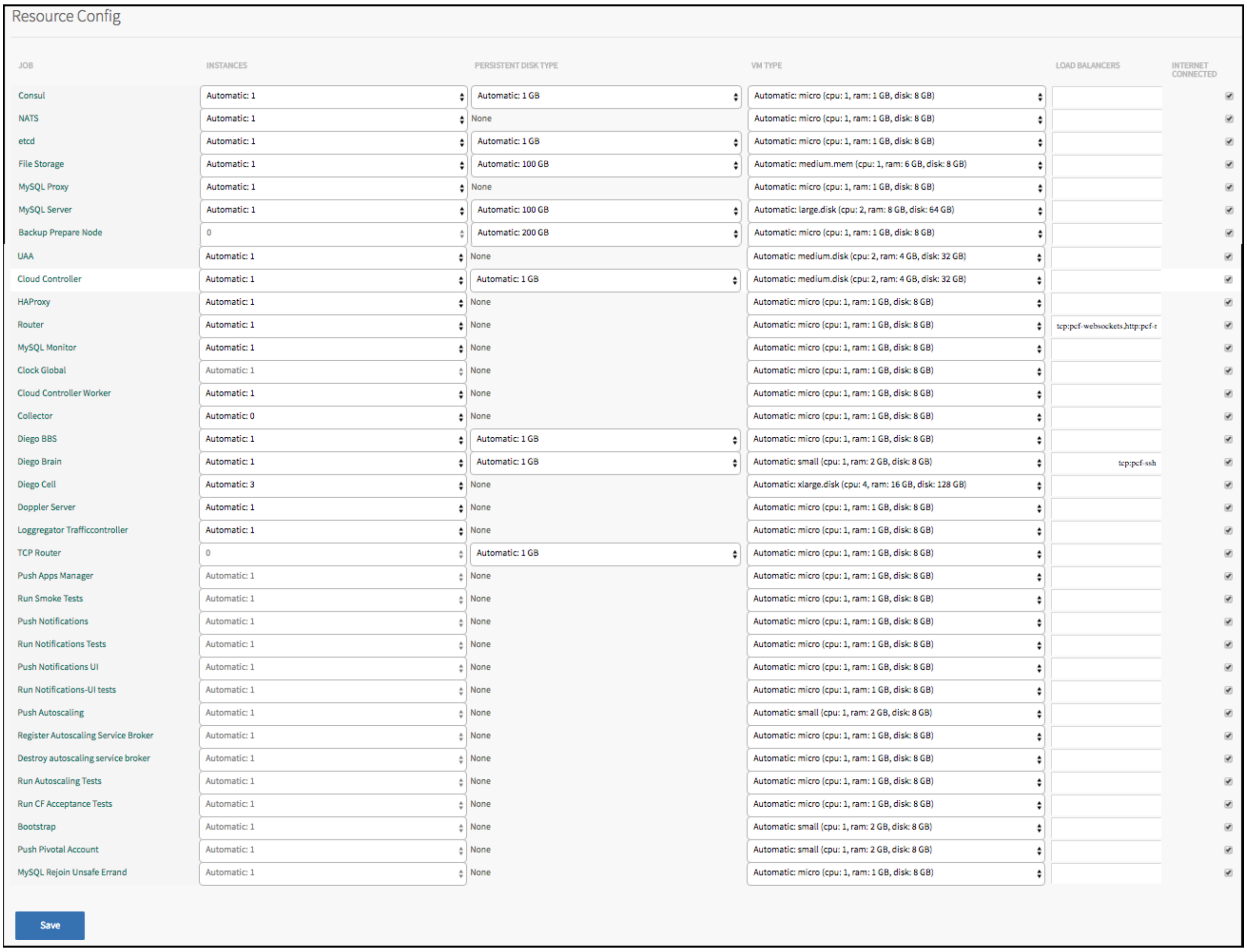
2. Record the name of your SSH load balancer and your TCP WebSockets load balancer, `MY-PCF-wss-logs` and `MY-PCF-ssh-proxy`.

3. Click your HTTP(S) load balancer, `MY-PCF-global-pcf`.



4. Under **Backend services**, record the name of the backend service of the HTTP(S) load balancer, `MY-PCF-http-lb-backend`.

5. In the PAS tile, click **Resource Config**.



6. Under the **LOAD BALANCERS** column of the **Router** row, enter a comma-delimited list consisting of the name of your TCP WebSockets load balancer and the name of your HTTP(S) load balancer backend with the protocol prepended. For example, `tcp:MY-PCF-wss-logs,http:MY-PCF-http-lb-backend`.

> **Note:** Do not add a space between key/value pairs in the `LOAD BALANCER` field or it will fail.

> **Note:** If you are using HAProxy in your deployment, then enter the above load balancer values in the `LOAD BALANCERS` field of the

HAProxy row instead of the **Router** row. For a high availability configuration, scale up the HAProxy job to more than one instance.

7. If you have enabled TCP routing in the Networking pane and set up the TCP Load Balancer in GCP, add the name of your TCP load balancer, prepended with `tcp:`, to the **LOAD BALANCERS** column of the TCP Router row. For example, `tcp:pcf-tcp-router`.

8. Enter the name of you SSH load balancer depending on which release you are using.
   - **PAS**: Under the **LOAD BALANCERS** column of the **Diego Brain** row, enter the name of your SSH load balancer prepended with `tcp:`. For example, `tcp:MY-PCF-ssh-proxy`.
   - **Small Footprint Runtime**: Under the **LOAD BALANCERS** column of the **Control** row, enter the name of your SSH load balancer prepended with `tcp:`.

9. Verify that the **Internet Connected** checkbox for every job is unchecked. When preparing your GCP environment, you provisioned a Network Address Translation (NAT) box to provide Internet connectivity to your VMs instead of providing them with public IP addresses to allow the jobs to reach the Internet.

   > **Note**: If you want to provision a Network Address Translation (NAT) box to provide Internet connectivity to your VMs instead of providing them with public IP addresses, deselect the **Internet Connected** checkboxes. For more information about using NAT in GCP, see the GCP documentation.

10. Click **Save**.

Step 24: (Optional) Scale Down and Disable Resources

> **Note**: The **Resource Config** pane has fewer VMs if you are installing the Small Footprint Runtime.

> **Note**: The Small Footprint Runtime does not default to a highly available configuration. It defaults to the minimum configuration. If you want to make the Small Footprint Runtime highly available, scale the **Compute**, **Router**, and **Database** VMs to `3` instances and scale the **Control** VM to `2` instances.

Pivotal Application Service (PAS) defaults to a highly available resource configuration. However, you may need to perform additional procedures to make your deployment highly available. See the Zero Downtime Deployment and Scaling in CF and the Scaling Instances in PAS topics for more information.

If you do not want a highly available resource configuration, you must scale down your instances manually by navigating to the **Resource Config** section and using the drop-down menus under **Instances** for each job.

By default, PAS also uses an internal filestore and internal databases. If you configure PAS to use external resources, you can disable the corresponding system-provided resources in Ops Manager to reduce costs and administrative overhead.

Complete the following procedures to disable specific VMs in Ops Manager:

1. Click **Resource Config**.

2. If you configure PAS to use an external S3-compatible filestore, edit the following fields:
   - **File Storage**: Enter `0` in **Instances**.

3. If you selected **External** when configuring the UAA or System databases, edit the following fields:
   - **MySQL Proxy**: Enter `0` in **Instances**.
   - **MySQL Server**: Enter `0` in **Instances**.
   - **MySQL Monitor**: Enter `0` in **Instances**.
   - **Cloud Controller Database**: Enter `0` in **Instances**.
   - **UAA Database**: Enter `0` in **Instances**.

4. If you are not using HAProxy, enter `0` in the **Instances** field for **HAProxy**.

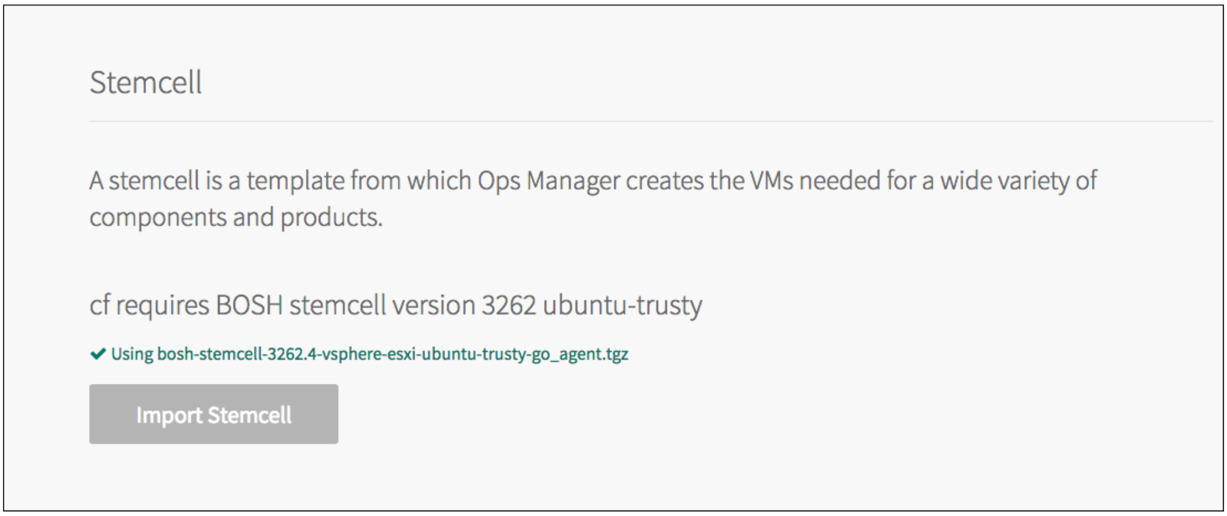
5. Click **Save**.

Step 25: Verify and Download Stemcell Version

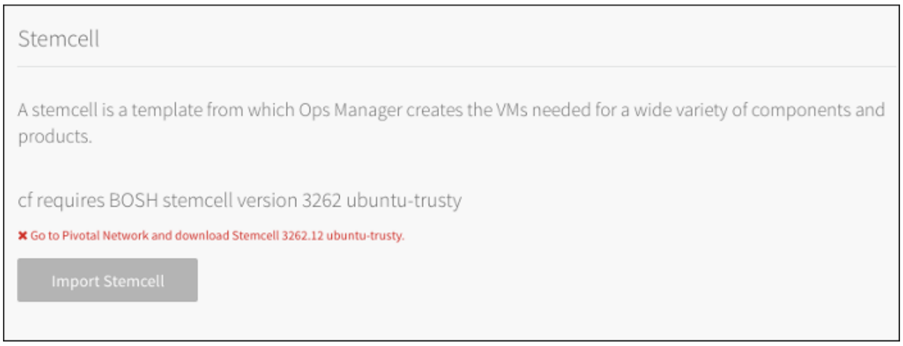
Verify whether Ops Manager is providing the stemcell version required by PAS. If the correct version is already present, you do not need to download a new stemcell.

1. In the PAS tile, select **Stemcell**.

2. Verify that the version indicated in the filename matches the version of stemcell required by PAS.
    - If PAS detects that a stemcell `.tgz` file is present in the Ops Manager Director VM at `/var/tempest/stemcells/`, the Stemcell screen displays filename information.



    - If PAS cannot detect a stemcell `.tgz` file, the following message displays:



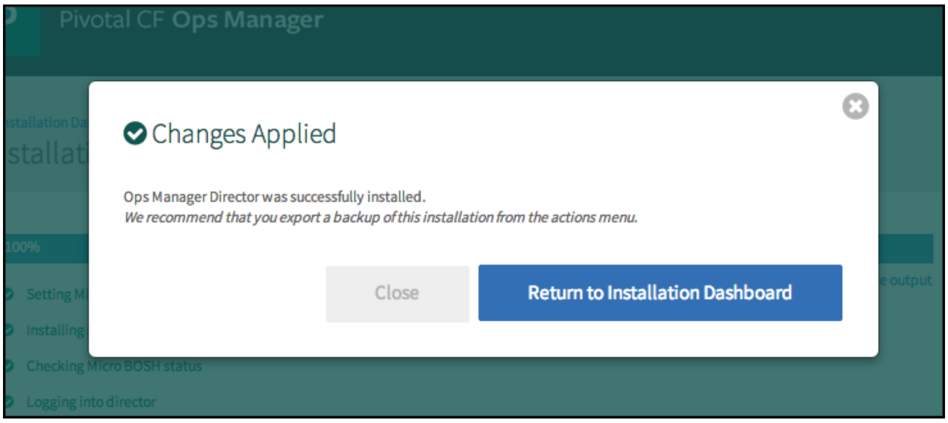
3. If the version of the stemcell file that is loaded does not match the required version listed in the Pivotal Network download page for PAS, or cannot be found by Ops Manager, perform the following steps to download and import a new stemcell file:

    a. Log in to the Pivotal Network and click **Stemcells**.
    b. Download the appropriate stemcell version targeted for your IaaS.
    c. In the **Stemcell** section of the PAS tile, click **Import Stemcell** to import the downloaded stemcell `.tgz` file.

Step 26: Complete the PAS Installation

1. Click the **Installation Dashboard** link to return to the Installation Dashboard.

2. Click **Apply Changes**.
   The install process generally requires a minimum of 90 minutes to complete. The image shows the Changes Applied window that displays when the installation process successfully completes.


Pivotal CF Ops Manager
Changes Applied
Ops Manager Director was successfully installed.
We recommend that you export a backup of this installation from the actions menu.
Close
Return to Installation Dashboard
100%
Checking Micro BOSH status
Logging into director

Installing PCF on GCP using Terraform

Complete the following procedures to install PCF on GCP using Terraform:

1. Preparing to Deploy PCF on GCP (Terraform)
2. Configuring Ops Manager Director on GCP (Terraform)
3. (Optional) Configuring a Shared VPC on GCP
4. (Optional) Installing the PCF IPsec Add-On
5. Deploying Pivotal Application Service (PAS) on GCP (Terraform)

Pivotal

Preparing to Deploy PCF on GCP with Terraform

Page last updated:

This guide describes the preparation steps required to install Pivotal Cloud Foundry (PCF) on Google Cloud Platform (GCP) using Terraform templates.

The Terraform template for PCF on GCP describes a set of GCP resources and properties. For more information about how Terraform creates resources in GCP, see the Google Cloud Provider topic on the Terraform site.

You may also find it helpful to review different deployment options in the Reference Architecture for Pivotal Cloud Foundry on GCP.

Prerequisites

In addition to fulfilling the prerequisites listed in the Installing Pivotal Cloud Foundry on GCP topic, ensure you have the following:

- The Terraform CLI
- The Google Cloud SDK
- In your GCP project, enable the following APIs:
  - Identity and Access Management
  - Cloud Resource Manager
  - Cloud DNS
  - Cloud SQL API

Step 1: Obtain a GCP Service Account Key File

To use the Terraform templates to create the necessary infrastructure resources for PCF, you need a service account key file. Follow the procedure that corresponds to your use case:

- **I already have a service account I want to use**
  1. Navigate to the GCP console
  2. Select **IAM** and locate your service account.
  3. From the **Options** column, open the dropdown menu and click **Create Key**.
- **I want to create a new service account**
  1. Open a terminal window.
  2. Create a service account using the gcloud CLI:

     ```
     $ gcloud iam service-accounts create ACCOUNT-NAME
     ```

  3. Create a key file for your service account:

     ```
     $ gcloud iam service-accounts keys create "terraform.key.json" --iam-account " some-account-name@yourproject.iam.gserviceaccount.com"
     ```

  4. Bind the service account to your project and give it the owner role:

     ```
     $ gcloud projects add-iam-policy-binding PROJECT_ID --member 'serviceAccount: some-account-name@PROJECT_ID.iam.gserviceaccount.com' --role 'roles/owner'
     ```

Step 2: Download and Edit the Terraform Variables File

Before you can run Terraform commands to create infrastructure resources, you must fill out a template variables file.

1. Navigate to the Pivotal Application Service (PAS) release on Pivotal Network.

2. Download the GCP Terraform zip file.

3. Extract the contents of the zip file and place the folder in your `workspace` directory on your local machine.

4. From a terminal window, navigate to the folder:

```
$ cd ~/workspace/TERRAFORMING-GCP-FOLDER
```

5. Create a new file named `terraform.tfvars`.

```
$ touch terraform.tfvars
```

6. Open the `terraform.tfvars` file and paste in the following contents:

```
env_name         = "YOUR-ENVIRONMENT-NAME"
opsman_image_url = "YOUR-OPS-MAN-IMAGE-URL"
region           = "YOUR-GCP-REGION"
zones            = ["YOUR-AZ-1", "YOUR-AZ-2", "YOUR-AZ-3"]
project          = "YOUR-GCP-PROJECT"
dns_suffix       = "YOUR-DNS-SUFFIX"

ssl_cert = <<SSL_CERT
-----BEGIN CERTIFICATE-----
YOUR-CERTIFICATE
-----END CERTIFICATE-----
SSL_CERT

ssl_cert_private_key = <<SSL_KEY
-----BEGIN EXAMPLE RSA PRIVATE KEY-----
YOUR-PRIVATE-KEY
-----END EXAMPLE RSA PRIVATE KEY-----
SSL_KEY

service_account_key = <<SERVICE_ACCOUNT_KEY
YOUR-KEY-JSON
SERVICE_ACCOUNT_KEY
```

7. Edit the values in the file according to the table below:

Value to replace	Guidance
`YOUR-ENVIRONMENT-NAME`	Enter a name to use to identify resources in GCP. Terraform prepends the names of the resources it creates with this environment name. Example: `pcf`.
`YOUR-OPS-MAN-IMAGE-URL`	Enter the source URL of the Ops Manager image you want to boot. You can find this URL in the PDF included with the Ops Manager release on Pivotal Network.
`YOUR-GCP-REGION`	Enter the name of the GCP region in which you want Terraform to create resources. Example: `us-central1`.

`YOUR-AZ-1`
`YOUR-AZ-2`
`YOUR-AZ-3`
	Enter three availability zones from your region. Example: `us-central1-a`, `us-central1-b`, `us-central1-c`.

`YOUR-GCP-PROJECT`	Enter the name of the GCP project in which you want Terraform to create resources.
`YOUR-DNS-SUFFIX`	Enter a domain name to use as part of the system domain for your PCF deployment. Terraform creates DNS records in GCP using `YOUR-ENVIRONMENT-NAME` and `YOUR-DNS-SUFFIX`. For example, if you enter `example.com` for your DNS suffix and have `pcf` as your environment name, Terraform will create DNS records at `pcf.example.com`.
`YOUR-CERTIFICATE`	Enter a certificate to use for HTTP load balancing. For production environments, use a certificate from a Certificate Authority (CA). For test environments, you can use a self-signed certificate. Your certificate must specify your system domain as the common name. Your system domain is `YOUR-ENVIRONMENT-NAME.YOUR-DNS-SUFFIX`. It also must include the following subdomains: `*.sys.YOUR-SYSTEM-DOMAIN`, `*.login.sys.YOUR-SYSTEM-DOMAIN`, `*.uaa.sys.YOUR-SYSTEM-DOMAIN`, `*.apps.YOUR-SYSTEM-DOMAIN`.
`YOUR-PRIVATE-KEY`	Enter a private key for the certificate you entered.
`YOUR-KEY-JSON`	Enter the contents of your service account key file. This file is in JSON format.

Step 3: Add Optional Variables

Complete this step if you want to do any of the following:

- Change the default CIDR ranges
- Deploy the Isolation Segment tile
- Use an external Google Cloud SQL database
- Use external Google Storage buckets

In your `terraform.tfvars` file, specify the appropriate variables from the sections below.

> **Note**: You can see the configurable options by opening the `variables.tf` file and looking for variables with default values.

CIDR Ranges for Subnets

If you want to change the CIDR ranges for the management, PAS, or services networks that Terraform creates, add the following variables to your `terraform.tfvars` file, replacing `YOUR-MANAGEMENT-CIDR` , `YOUR-PAS-CIDR` and `YOUR-SERVICES-CIDR` with your desired values.

```
management_cidr = YOUR-MANAGEMENT-CIDR
pas_cidr = YOUR-PAS-CIDR
services_cidr = YOUR-SERVICES-CIDR
```

Isolation Segments

If you plan to deploy the Isolation Segment tile, add the following variables to your `terraform.tfvars` file, replacing `YOUR-CERTIFICATE` and `YOUR-PRIVATE-KEY` with a certificate and private key. This causes terraform to create an additional HTTP load balancer across three availability zones to use for the Isolation Segment tile.

```
isolation_segment = true
iso_seg_ssl_cert = <<ISO_SEG_SSL_CERT
-----BEGIN CERTIFICATE-----
YOUR-CERTIFICATE
-----END CERTIFICATE-----
ISO_SEG_SSL_CERT
iso_seg_ssl_cert_private_key = <<ISO_SEG_SSL_KEY
-----BEGIN EXAMPLE RSA PRIVATE KEY-----
YOUR-PRIVATE-KEY
-----END EXAMPLE RSA PRIVATE KEY-----
ISO_SEG_SSL_KEY
```

External Database

1. If you want to use an external Google Cloud SQL database for Ops Manager and Pivotal Application Service (PAS), add the following to your `terraform.tfvars` file:

   ```
   external_database = true
   ```

2. If you want to specify a single host from which users can connect to the Ops Manger and PAS database, add the following variables to your `terraform.tfvars` file. Replace `HOST_IP_ADDRESS` with your desired IP addresses.

   ```
   opsman_sql_db_host = HOST_IP_ADDRESS
   pas_sql_db_host = HOST_IP_ADDRESS
   ```

External Storage Buckets

If you want to use Google Cloud Storage buckets for the PAS Cloud Controller, add the following to your `terraform.tfvars` file:

```
pas_sql_db_host = true
```

Step 4: Create GCP Resources with Terraform

Follow these steps to use the Terraform CLI to create resources on GCP:

1. From the directory that contains the Terraform files, run the following command to initialize the directory based on the information you specified in the `terraform.tfvars` file.

   ```
   $ terraform init
   ```

2. Run the following command. It will create the execution plan for Terraform.

   ```
   $ terraform plan -out=plan
   ```

3. Run the following command. It will execute the plan from the previous step. It may take several minutes for Terraform to create all the resources in GCP.

   ```
   $ terraform apply plan
   ```

Step 5: Create DNS Record

1. In a browser, navigate to the DNS provider for the DNS suffix you entered in your `terraform.tfvars` file.

2. Create a new NS (Name server) record for your PCF system domain. Your system domain is `YOUR-ENVIRONMENT-NAME.YOUR-DNS-SUFFIX`.

   a. In this record, enter the name servers included in `env_dns_zone_name_servers` from your Terraform output.

What to Do Next

Proceed to the next step in the deployment, [Configuring Ops Manager Director on GCP (Terraform)](#).

Configuring Ops Manager Director on GCP (Terraform)

Page last updated:

This topic describes how to configure the Ops Manager Director for Pivotal Cloud Foundry (PCF) on Google Cloud Platform (GCP) after Preparing to Deploy PCF on GCP with Terraform.

> **Note**: You can also perform the procedures in this topic using the Ops Manager API. For more information, see the Using the Ops Manager API topic.

Prerequisite

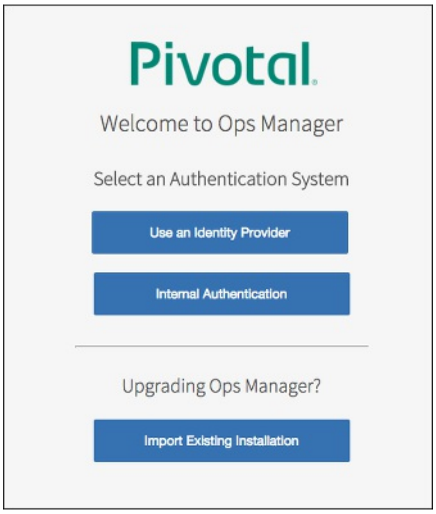
To complete the procedures in this topic, you must have access to the output from when you ran `terraform apply` to create resources for this deployment. You can view this output at any time by running `terraform output`. You use the values in your Terraform output to configure the Ops Manager Director tile.

Step 1: Access Ops Manager

1. In a web browser, navigate to the fully qualified domain name (FQDN) of the Ops Manager director. Use the `ops_manager_dns` value from running `terraform output`.

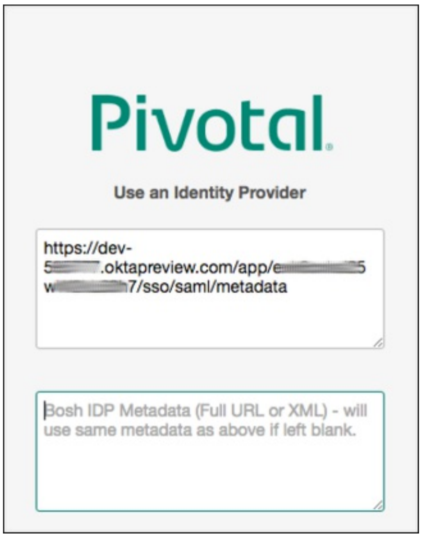
   > **Note**: In order to set up Ops Manager authentication correctly, Pivotal recommends using a Fully Qualified Domain Name (FQDN) to access Ops Manager. Using an ephemeral IP address to access Ops Manager can cause authentication errors upon subsequent access.

2. When Ops Manager starts for the first time, you must choose one of the following:
   - Use an Identity Provider: If you use an Identity Provider, an external identity server maintains your user database.
   - Internal Authentication: If you use Internal Authentication, PCF maintains your user database.



Use an Identity Provider (IdP)

1. Log in to your IdP console and download the IdP metadata XML. Optionally, if your IdP supports metadata URL, you can copy the metadata URL instead of the XML.



2. Copy the IdP metadata XML or URL to the Ops Manager **Use an Identity Provider** log in page.

> **Note**: The same IdP metadata URL or XML is applied for the BOSH Director. If you use a separate IdP for BOSH, copy the metadata XML or URL from that IdP and enter it into the BOSH IdP Metadata text box in the Ops Manager log in page.

3. Enter your **Decryption passphrase**. Read the **End User License Agreement**, and select the checkbox to accept the terms.

4. Your Ops Manager log in page appears. Enter your username and password. Click **Login**.

5. Download your SAML Service Provider metadata (SAML Relying Party metadata) by navigating to the following URLs:
   - **5a.** Ops Manager SAML service provider metadata: `https://OPS-MAN-FQDN:443/uaa/saml/metadata`
   - **5b.** BOSH Director SAML service provider metadata: `https://BOSH-IP-ADDRESS:8443/saml/metadata`

> **Note**: To retrieve your `BOSH-IP-ADDRESS`, navigate to the **Ops Manager Director** tile > **Status** tab. Record the **Ops Manager Director** IP address.

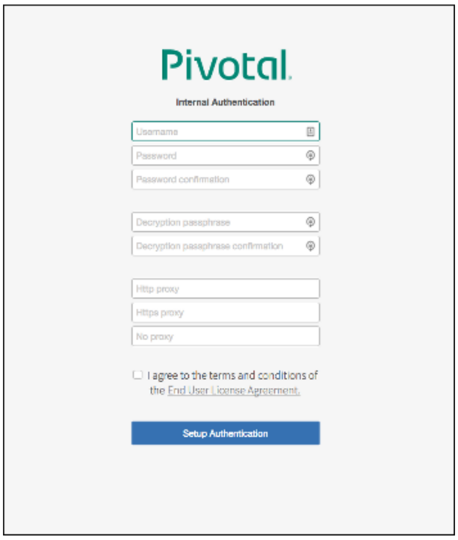
6. Configure your IdP with your SAML Service Provider metadata. Import the Ops Manager SAML provider metadata from Step 5a above to your IdP. If your IdP does not support importing, provide the values below.
   - **Single sign on** URL: `https://OPS-MAN-FQDN:443/uaa/saml/SSO/alias/OPS-MAN-FQDN`
   - **Audience URI** (SP Entity ID): `https://OP-MAN-FQDN:443/uaa`
   - **Name ID**: Email Address
   - SAML authentication requests are always signed

7. Import the BOSH Director SAML provider metadata from Step 5b to your IdP. If the IdP does not support an import, provide the values below.
   - **Single sign on** URL: `https://BOSH-IP:8443/saml/SSO/alias/BOSH-IP`
   - **Audience URI** (SP Entity ID): `https://BOSH-IP:8443`
   - **Name ID**: Email Address
   - SAML authentication requests are always signed

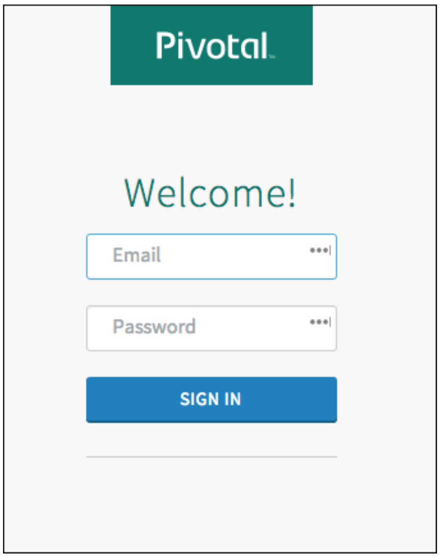
8. Return to the **Ops Manager Director** tile, and continue with the configuration steps below.

Internal Authentication

1. When redirected to the **Internal Authentication** page, you must complete the following steps:
   - Enter a **Username**, **Password**, and **Password confirmation** to create an Admin user.
   - Enter a **Decryption passphrase** and the **Decryption passphrase confirmation**. This passphrase encrypts the Ops Manager datastore, and is not recoverable if lost.
   - If you are using an **HTTP proxy** or **HTTPS proxy**, follow the instructions in the Configuring Proxy Settings for the BOSH CPI topic.
   - Read the **End User License Agreement**, and select the checkbox to accept the terms.
   - Click **Setup Authentication**.



2. Log in to Ops Manager with the Admin username and password that you created in the previous step.

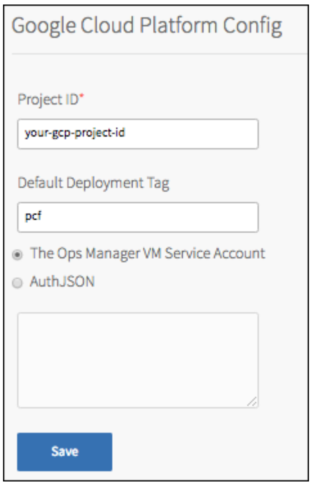


Step 2: Google Cloud Platform Config

1. Click the **Google Cloud Platform** tile within the **Installation Dashboard**.

2. Select **Google Config**. Complete the following fields:
   - **Project ID**: Enter the value of `project` from your `terraform.tfvars` file.
   - **Default Deployment Tag**: Enter the value of `env_name` from your `terraform.tfvars` file.
   - Select **AuthJSON** and in the field below enter the contents of the JSON file for you service account key.

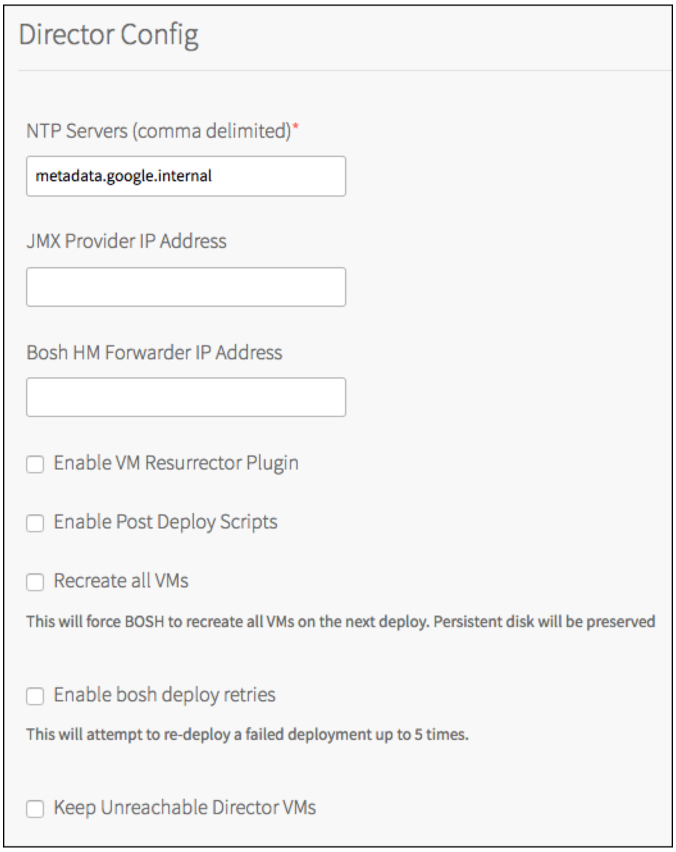
> **Note**: As an alternative, you can select **The Ops Manager VM Service Account** option to use the service account automatically created by GCP for the Ops Manager VM.



3. Click **Save**.

Step 3: Director Config Page

1. Select **Director Config** to open the **Director Config** page.



2. In the **NTP Servers (comma delimited)** field, enter `metadata.google.internal`.

> **Note:** To resolve `metadata.google.internal` as the NTP server hostname, you must provide the two IP addresses for DNS configuration as described in [Step 5: Create Networks Page](#) of this procedure.

3. Leave the **JMX Provider IP Address** field blank.

> **Note:** Starting from PCF v2.0, BOSH-reported system metrics are available in the Loggregator Firehose by default. Therefore, if you continue to use PCF JMX Bridge for consuming them outside of the Firehose, you may receive duplicate data. To prevent this, leave the **JMX Provider IP Address** field blank. For additional guidance, see [BOSH System Metrics Available in Loggregator Firehose](#).

4. Leave the **Bosh HM Forwarder IP Address** field blank.

> **Note**: Starting from PCF v2.0, BOSH-reported system metrics are available in the Loggregator Firehose by default. Therefore, if you continue to use the BOSH HM Forwarder for consuming them, you may receive duplicate data. To prevent this, leave the **Bosh HM Forwarder IP Address** field blank. For additional guidance, see BOSH System Metrics Available in Loggregator Firehose.

5. Select the **Enable VM Resurrector Plugin** checkbox to enable the Ops Manager Resurrector functionality and increase Pivotal Application Service (PAS) availability.

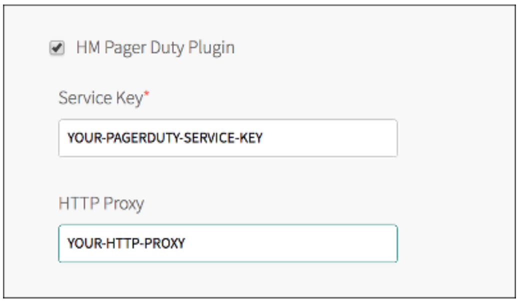
6. (Optional) Select **Enable Post Deploy Scripts** to run a post-deploy script after deployment. This script allows the job to execute additional commands against a deployment.

7. (Optional) Select **Recreate all VMs** to force BOSH to recreate all VMs on the next deploy. This process does not destroy any persistent disk data.

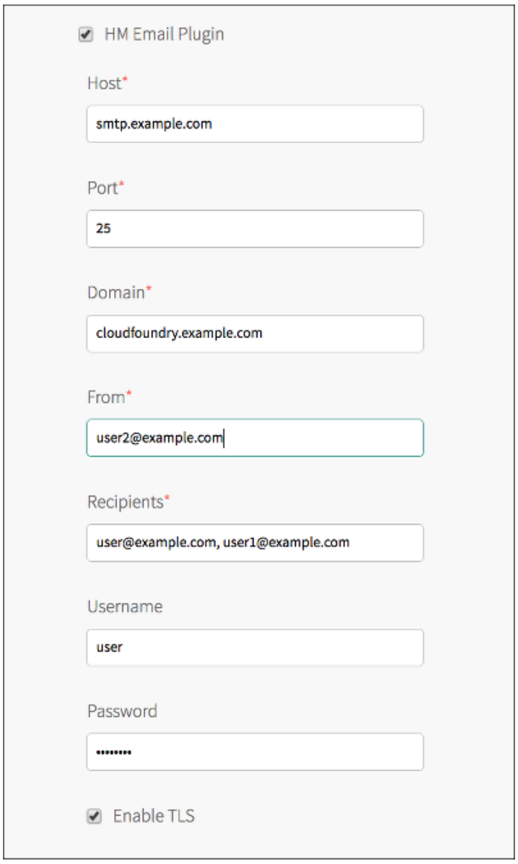
8. Select **Enable bosh deploy retries** for Ops Manager to retry failed BOSH operations up to five times.

9. (Optional) Select **Keep Unreachable Director VMs** if you want to preserve Ops Manager Director VMs after a failed deployment for troubleshooting purposes.

10. (Optional) Select **HM Pager Duty Plugin** to enable Health Monitor integration with PagerDuty.



    - **Service Key**: Enter your API service key from PagerDuty.
    - **HTTP Proxy**: Enter an HTTP proxy for use with PagerDuty.



11. (Optional) Select **HM Email Plugin** to enable Health Monitor integration with email.
    - **Host**: Enter your email hostname.

- **Port**: Enter your email port number.
- **Domain**: Enter your domain.
- **From**: Enter the address for the sender.
- **Recipients**: Enter comma-separated addresses of intended recipients.
- **Username**: Enter the username for your email server.
- **Password**: Enter the password for your email server.
- **Enable TLS**: Select this checkbox to enable Transport Layer Security.

12. For **Blobstore Location**, select **Internal**.

13. For **Database Location**, if you configured your `terraform.tfvars` file to create an external database for Ops Manager, select **External** and complete the fields below. Otherwise, select **Internal**.
    - **Host**: Enter the value of `sql_db_ip` from your Terraform output.
    - **Port**: Enter `3306`.
    - **Username**: Enter the value of `opsman_sql_username` from your Terraform output.
    - **Password**: Enter the value of `opsman_sql_password` from your Terraform output.
    - **Database**: Enter the value of `opsman_sql_db_name` from your Terraform output.

14. (Optional) Modify the **Director Workers** value, which sets the number of workers available to execute Director tasks. This field defaults to `5`.

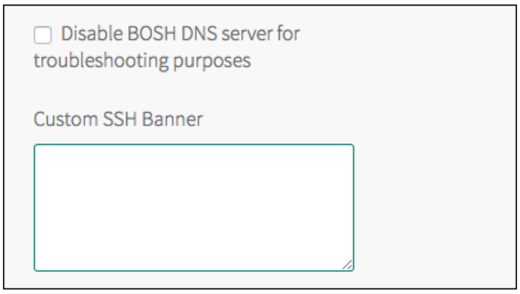
15. (Optional) **Max Threads** sets the maximum number of threads that the Ops Manager Director can run simultaneously. Pivotal recommends that you leave the field blank to use the default value, unless doing so results in rate limiting or errors on your IaaS.

16. (Optional) To add a custom URL for your Ops Manager Director, enter a valid hostname in **Director Hostname**. You can also use this field to configure a load balancer in front of your Ops Manager Director.

17. (Optional) To disable BOSH DNS, select the **Disable BOSH DNS server for troubleshooting purposes** checkbox. For more information about the BOSH DNS service discovery mechanism, see BOSH DNS Service Discovery for Application Containers in the Pivotal Application Service (PAS) Release Notes.

    Breaking Change: Do not disable BOSH DNS without consulting Pivotal Support. For more information about disabling BOSH DNS, see Disabling or Opting Out of BOSH DNS in PCF in the Pivotal Knowledge Base.

18. (Optional) To set a custom banner that users see when logging in to the Director using SSH, enter text in the **Custom SSH Banner** field.



19. Click **Save**.

Step 4: Create Availability Zones Page

Note: Pivotal recommends at least three availability zones for a highly available installation of PAS. For an example of a three availability zone deployment, see Reference Architecture for Pivotal Cloud Foundry on GCP.

1. Select **Create Availability Zones**.

2. Use the **Add** button to add three availability zones corresponding to those listed in the `azs` field in your terraform output.

3. Click **Save**.

Step 5: Create Networks Page

1. Select **Create Networks**.

2. Make sure **Enable ICMP checks** is not selected. GCP routers do not respond to ICMP pings.

3. Use the **Add Network** button to create the following three networks:

   Note: To use a shared VPC network, enter the shared VPC host project name before the network name in the format `VPC-PROJECT-NAME/NETWORK-NAME/SUBNET-NAME/REGION-NAME`. For example, `vpc-project/opsmgr/central/us-central1`. For more information, see Configuring a Shared VPC on GCP.

   Note: The `169.254.169.254` address points to the metadata server that hosts metadata for GCP instances. The `8.8.8.8` corresponds to Google's public DNS server. Using both addresses provides PCF with the ability to reach external DNS from app containers, but also keeps NTP working in the event that a VM does not have access to the Internet.

<table>
<tr><th>Network Name</th><th>Configuration</th></tr>
<tr><td><code>management</code></td><td>
<table>
<tr><td>Name</td><td><code>management</code></td></tr>
<tr><td>Service Network</td><td>Leave this checkbox unselected.</td></tr>
<tr><td>Google Network Name</td><td>Use the <code>network_name</code>, <code>management_subnet_name</code>, and <code>region</code> fields from your Terraform output to enter the name of the management network created by Terraform.
The format is: <code>network_name/management_subnet_name/region</code></td></tr>
<tr><td>CIDR</td><td>Enter the value of <code>management_subnet_cidrs</code> from your Terraform output.</td></tr>
<tr><td>Reserved IP Ranges</td><td>Enter the first <code>.1</code> through <code>.9</code> addresses from the CIDR. For example, if the CIDR is <code>192.168.101.0/26</code>, enter the range <code>192.168.101.1-192.168.101.9</code>.</td></tr>
<tr><td>DNS</td><td>Enter <code>169.254.169.254, 8.8.8.8</code>.</td></tr>
<tr><td>Gateway</td><td>Enter the value of <code>management_subnet_gateway</code> from your Terraform output.</td></tr>
<tr><td>Availability Zones</td><td>Select all three availability zones.</td></tr>
</table>
</td></tr>
<tr><td><code>pas</code></td><td>
<table>
<tr><td>Name</td><td><code>pas</code></td></tr>
<tr><td>Service Network</td><td>Leave this checkbox unselected.</td></tr>
<tr><td>Google Network Name</td><td>Use the <code>network_name</code>, <code>pas_subnet_name</code>, and <code>region</code> fields from your Terraform output to enter the name of the PAS network created by Terraform.
The format is: <code>network_name/pas_subnet_name/region</code></td></tr>
<tr><td>CIDR</td><td>Enter the value of <code>pas_subnet_cidrs</code> from your Terraform output.</td></tr>
<tr><td>Reserved IP Ranges</td><td>Enter the first <code>.1</code> through <code>.9</code> addresses from the CIDR. For example, if the CIDR is <code>192.168.16.0/22</code>, enter the range <code>192.168.16.1-192.168.16.9</code>.</td></tr>
<tr><td>DNS</td><td>Enter <code>169.254.169.254, 8.8.8.8</code>.</td></tr>
<tr><td>Gateway</td><td>Enter the value of <code>pas_subnet_gateway</code> from your Terraform output.</td></tr>
<tr><td>Availability Zones</td><td>Select all three availability zones.</td></tr>
</table>
</td></tr>
</table>

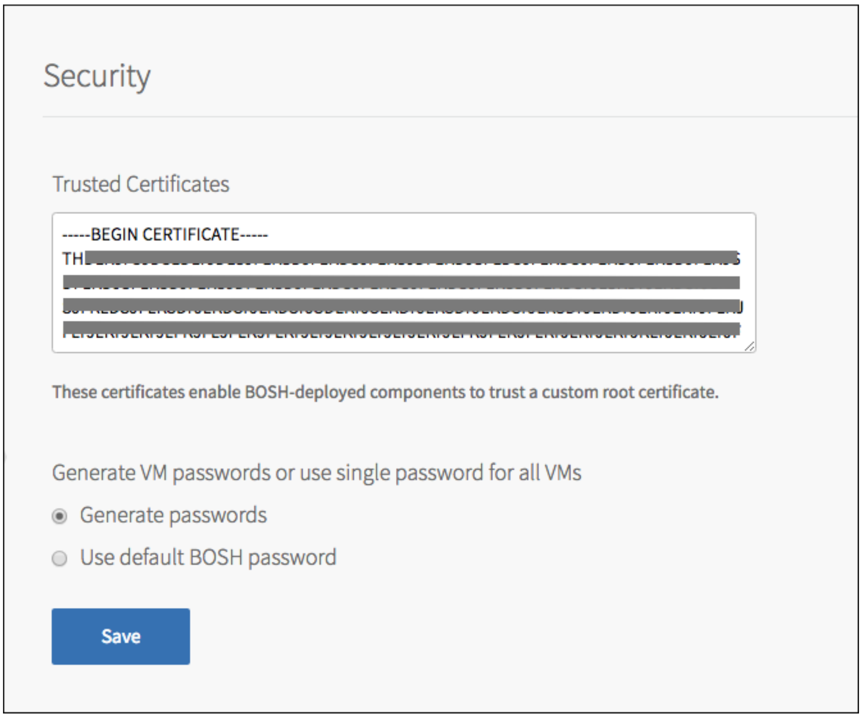
`services`	Name	`services`
	Service Network	Select the checkbox.
	Google Network Name	Use the `network_name` , `services_subnet_name` , and `region` fields from your Terraform output to enter the name of the services network created by Terraform. The format is: `network_name/services_subnet_name/region`
	CIDR	Enter the value of `services_subnet_cidrs` from your Terraform output.
	Reserved IP Ranges	Enter the first `.1` through `.9` addresses from the CIDR. For example, if the CIDR is `192.168.16.0/22` , enter the range `192.168.20.1-192.168.20.9` .
	DNS	Enter `169.254.169.254, 8.8.8.8` .
	Gateway	Enter the value of `services_subnet_gateway` from your Terraform output.
	Availability Zones	Select all three availability zones.

Step 6: Assign AZs and Networks Page

1. Select **Assign AZs and Networks**.
2. Use the drop-down menu to select a **Singleton Availability Zone**. The Ops Manager Director installs in this Availability Zone.
3. Under **Network**, select the `management` network for your Ops Manager Director.
4. Click **Save**.

Step 7: Security Page

1. Select **Security**.



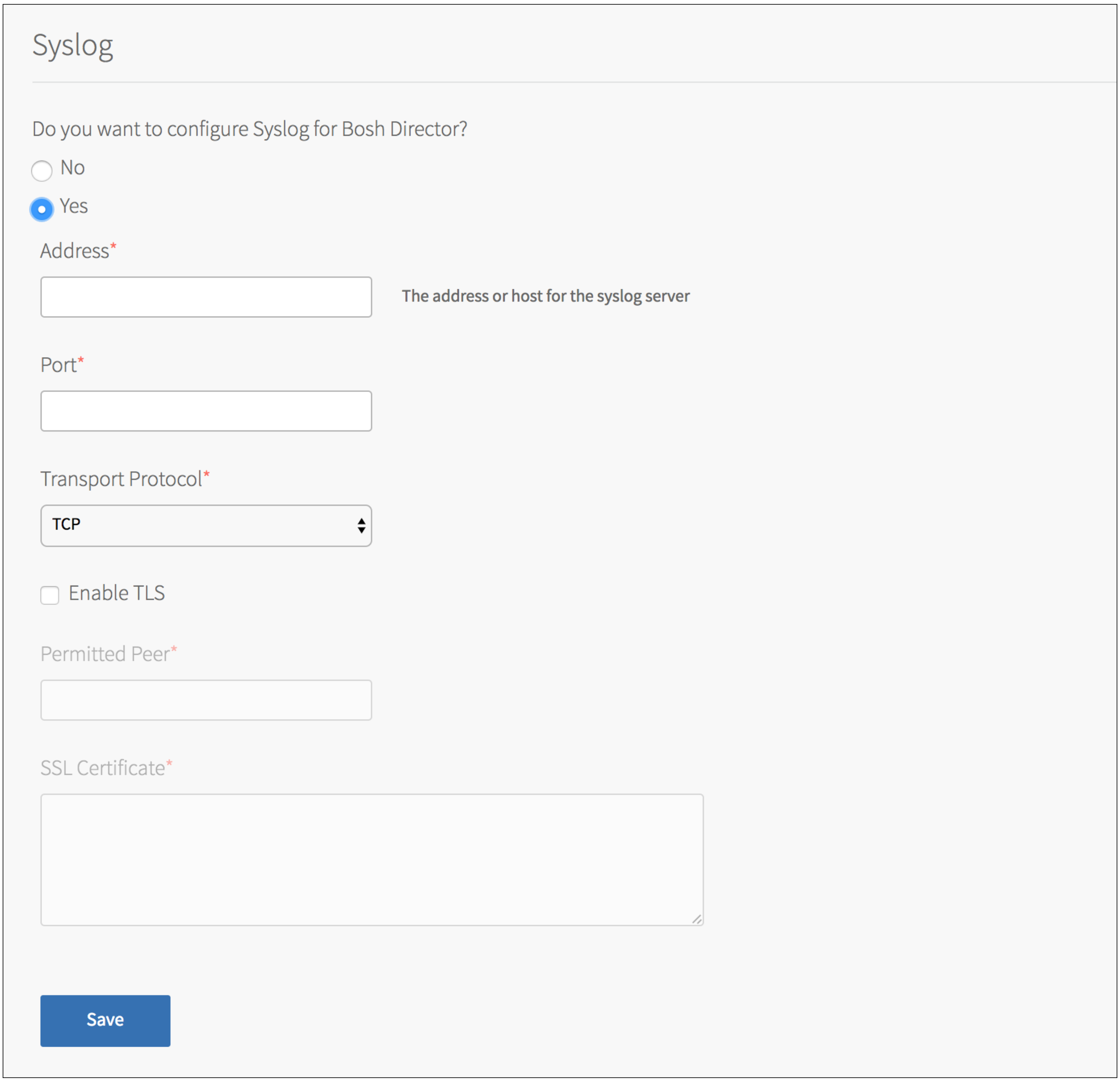
2. In **Trusted Certificates**, enter a custom certificate authority (CA) certificate to insert into your organization's certificate trust chain. This feature enables all BOSH-deployed components in your deployment to trust a custom root certificate.
   - You do not need to enter anything in this field if you are using self-signed certificates.
   - If you want to use Docker Registries for running app instances in Docker containers, enter the certificate for your private Docker Registry in this field. See the Using Docker Registries topic for more information.

3. Choose **Generate passwords** or **Use default BOSH password**. Pivotal recommends that you use the **Generate passwords** option for greater security.

4. Click **Save**. To view your saved Director password, click the **Credentials** tab.

Step 8: Syslog Page

1. Select **Syslog**.



2. (Optional) To send BOSH Director system logs to a remote server, select **Yes**.

3. In the **Address** field, enter the IP address or DNS name for the remote server.

4. In the **Port** field, enter the port number that the remote server listens on.

5. In the **Transport Protocol** dropdown menu, select **TCP**, **UDP**, or **RELP**. This selection determines which transport protocol is used to send the logs to the remote server.

6. (Optional) Mark the **Enable TLS** checkbox to use TLS encryption when sending logs to the remote server.
   - In the **Permitted Peer** field, enter either the name or SHA1 fingerprint of the remote peer.
   - In the **SSL Certificate** field, enter the SSL certificate for the remote server.

7. Click **Save**.

Step 9: Resource Config Page

1. Select **Resource Config**.

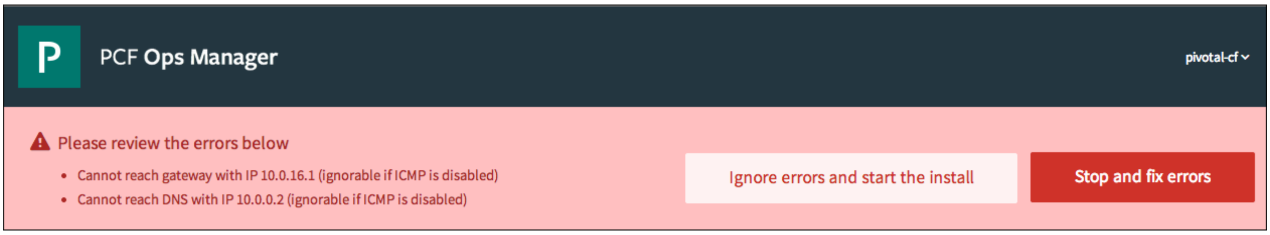
2. Verify that the **Internet Connected** checkbox for every job is checked. The terraform templates do not provision a Network Address Translation (NAT) box for Internet connectivity to your VMs so instead they will be provided with ephemeral public IP addresses to allow the jobs to reach the Internet.

> **Note:** If you want to provision a Network Address Translation (NAT) box to provide Internet connectivity to your VMs instead of providing them with public IP addresses, deselect the **Internet Connected** checkboxes. For more information about using NAT in GCP, see the GCP documentation.

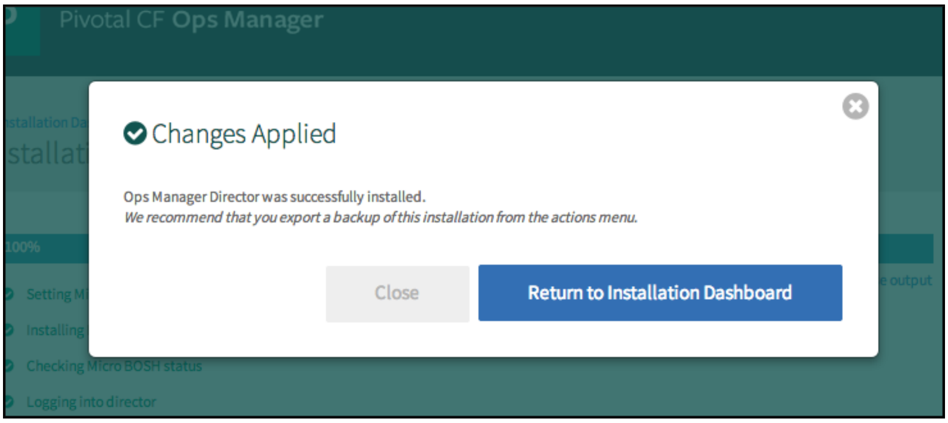
Step 10: Complete the Ops Manager Director Installation

1. Click the **Installation Dashboard** link to return to the Installation Dashboard.

2. Click **Apply Changes**. If the following ICMP error message appears, return to the Network Config screen, and make sure you have deselected the **Enable ICMP Checks** box. Then click **Apply Changes** again.



3. Ops Manager Director installs. This may take a few moments. When the installation process successfully completes, the **Changes Applied** window appears.



What to Do Next

After you complete this procedure, follow the instructions in the Deploying PAS on GCP topic.

Deploying PAS on GCP (Terraform)

Page last updated:

This topic describes how to install and configure Pivotal Application Service (PAS) on Google Cloud Platform (GCP).

Before beginning this procedure, ensure that you have successfully completed the Configuring Ops Manager Director on GCP (Terraform) topic.

> **Note**: If you plan to install the PCF IPsec add-on, you must do so before installing any other tiles. Pivotal recommends installing IPsec immediately after Ops Manager, and before installing the PAS tile.

Prerequisite

To complete the procedures in this topic, you must have access to the output from when you ran `terraform apply` to create resources for this deployment. You can view this output at any time by running `terraform output`. You use the values in your Terraform output to configure the Ops Manager Director tile.

Step 1: Download the PAS Tile

1. If you have not already downloaded PAS, log in to Pivotal Network, and click on **Pivotal Application Service**.
2. From the **Releases** drop-down, select the release to install and choose one of the following:
   a. Click **Pivotal Application Service** to download the PAS `.pivotal` file
   b. Click **PCF Small Footprint Runtime** to download the Small Footprint Runtime `.pivotal` file. For more information, see Getting Started with Small Footprint Runtime.

Step 2: Add PAS to Ops Manager

1. Navigate to the Pivotal Cloud Foundry Operations Manager Installation Dashboard.
2. Click **Import a Product** to add the PAS tile to Ops Manager. This may take a while depending on your connection speed.

   > **Tip**: After you import a tile to Ops Manager, you can view the latest available version of that tile in the Installation Dashboard by enabling the Pivotal Network API. For more information, refer to the Adding and Deleting Products topic.

3. On the left, click the plus icon next to the imported PAS product to add it to the Installation Dashboard.
4. Click the newly added PAS tile in the Installation Dashboard.

Step 3: Assign Availability Zones and Networks

1. Select **Assign AZ and Networks**. These are the Availability Zones that you create when configuring Ops Manager Director.
2. Select the first Availability Zone under **Place singleton jobs**. Ops Manager runs any job with a single instance in this Availability Zone.
3. Select all Availability Zones under **Balance other jobs**. Ops Manager balances instances of jobs with more than one instance across the Availability

Zones that you specify.

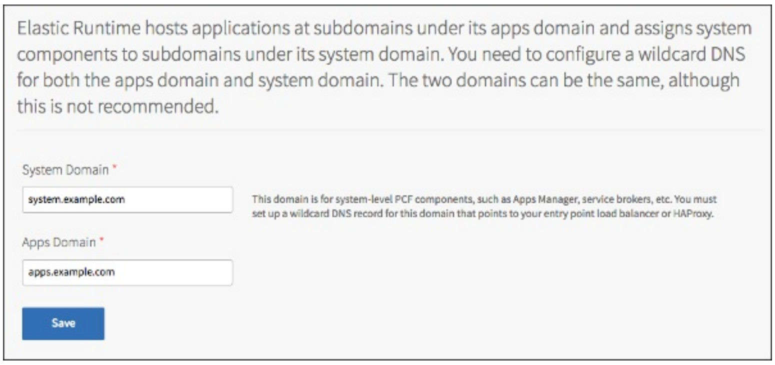
> **Note**: For production deployments, Pivotal recommends at least three Availability Zones for a highly available installation of PAS.

4. From the **Network** drop-down box, choose the `pas` network you created when configuring Ops Manager Director.

5. Click **Save**.

Step 5: Configure Domains

1. Select **Domains**.



2. Enter the system and application domains that were created by Terraform:
   - **System Domain**: Enter the value of `sys_domain` from your Terraform output.
   - **Apps Domain**: Enter the value of `apps_domain` from your Terraform output.

3. Click **Save**.

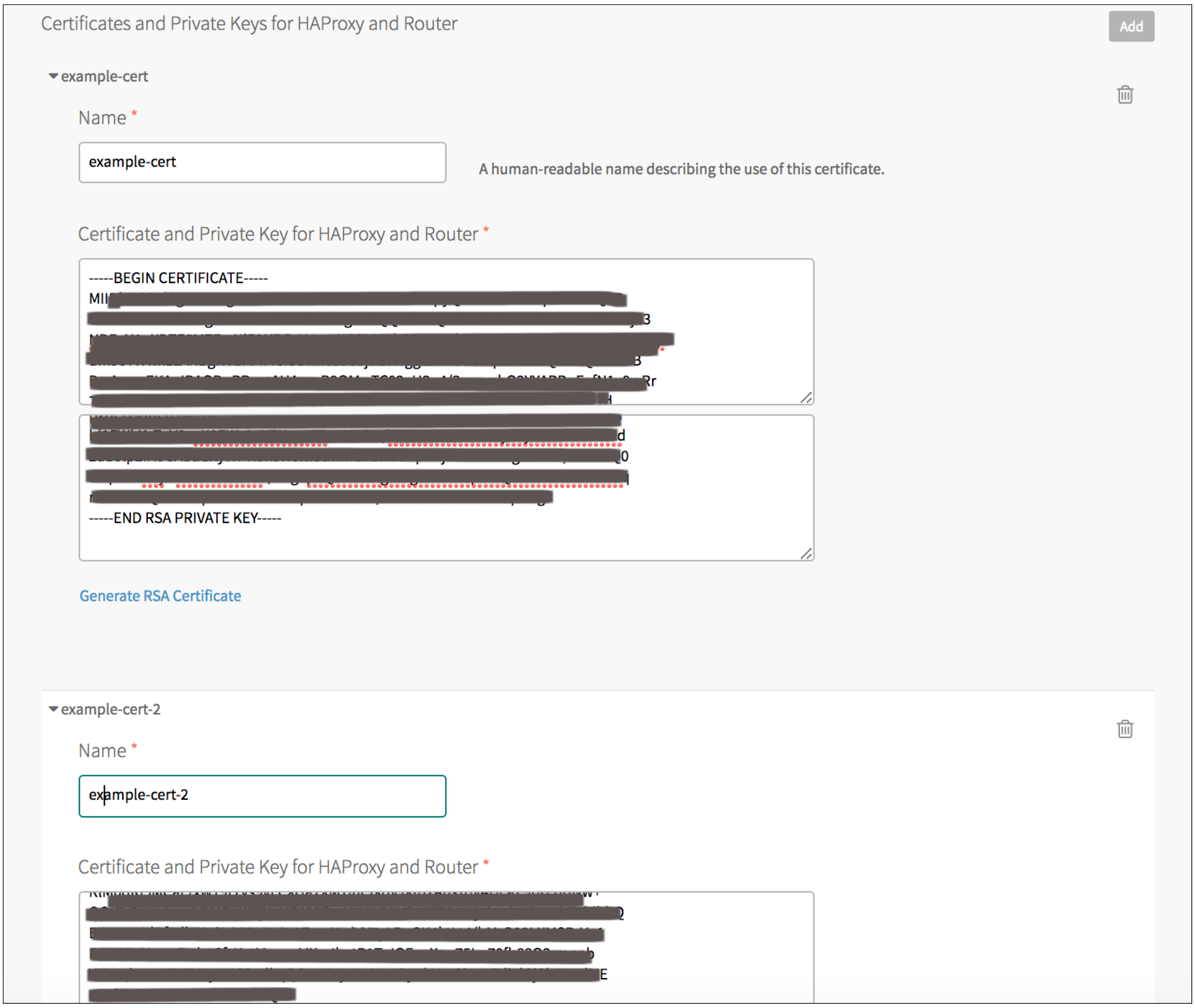
Step 6: Configure Networking

1. Select **Networking**.

2. Leave the **Router IPs**, **SSH Proxy IPs**, **HAProxy IPs**, and **TCP Router IPs** fields blank. You do not need to complete these fields when deploying PCF to GCP.

> **Note**: You specify load balancers in the **Resource Config** section of PAS later on in the installation process. See the Configure Load Balancers section of this topic for more information.

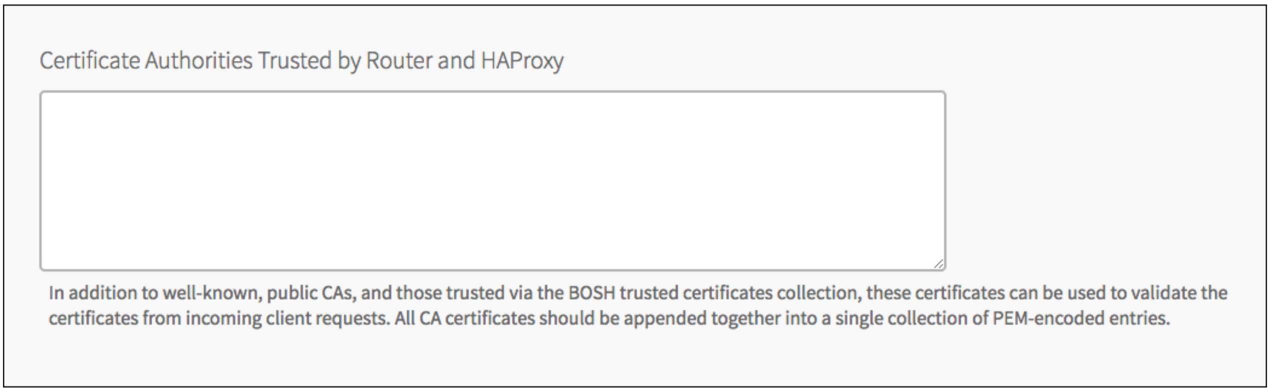
3. Under **Certificate and Private Key for HAProxy and Router**, you must provide an **SSL Certificate and Private Key**. Starting in PCF v.1.12, HAProxy and the Gorouter are enabled to receive TLS communication by default.



You can either provide a certificate signed by a Certificate Authority (CA) or click on the **Generate RSA Certificate** link to generate a self-signed certificate in Ops Manager. Ensure the certificate includes `*.YOUR-SYSTEM-DOMAIN`, `*.apps.YOUR-SYSTEM-DOMAIN`, and `*.sys.YOUR-SYSTEM-DOMAIN`.

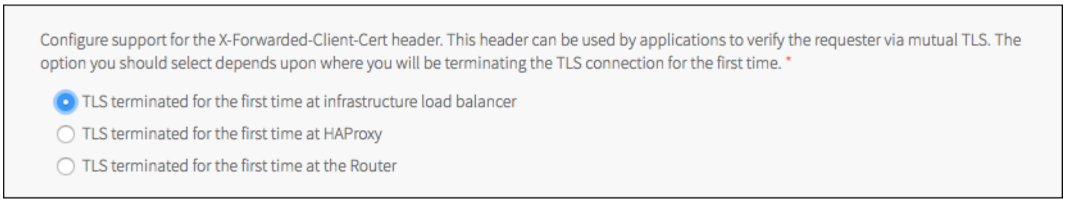
For details about generating certificates in Ops Manager for your wildcard system domains, see the Providing a Certificate for Your SSL/TLS Termination Point topic.

4. (Optional) When validating client requests using mutual TLS, the Gorouter trusts multiple certificate authorities (CAs) by default. If you want to configure the Gorouter and HAProxy to trust additional CAs, enter your CA certificates under **Certificate Authorities Trusted by Router and HAProxy**. All CA certificates should be appended together into a single collection of PEM-encoded entries.



5. In the **Minimum version of TLS supported by HAProxy and Router** field, select the minimum version of TLS to use in HAProxy and Router communications. HAProxy and Router use TLS v1.2 by default. If you need to accommodate clients that use an older version of TLS, select a lower minimum version. For a list of TLS ciphers supported by the Gorouter, see Securing Traffic into Cloud Foundry.

6. Under **Configure support for the X-Forwarded-Client-Cert header**, configure PCF handles `x-forwarded-client-cert` (XFCC) HTTP headers based on where TLS is terminated for the first time in your deployment.

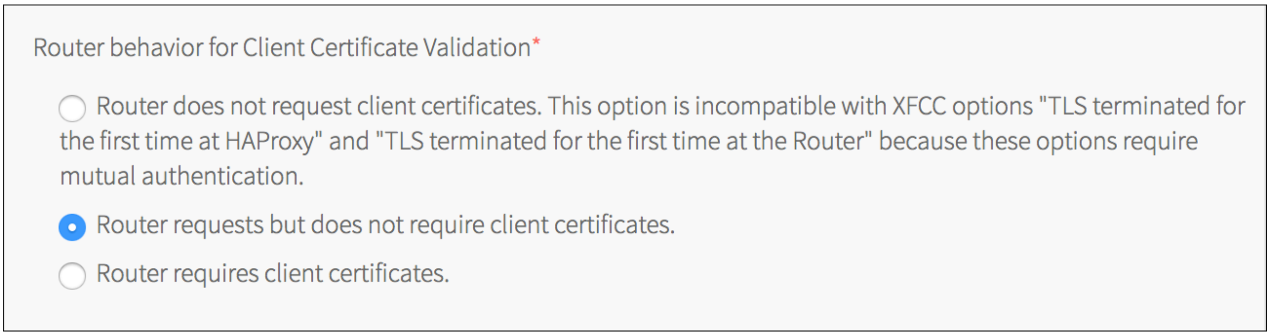


The following table indicates which option to choose based on your deployment layout.

If your deployment is configured as follows:	Then select the following option:	Additional notes:
◦ The Load Balancer is terminating TLS, and ◦ Load balancer is configured to put the client certificate from a mutual authentication TLS handshake into the X-Forwarded-Client-Cert HTTP header	TLS terminated for the first time at infrastructure load balancer (default).	Both HAProxy and the Gorouter forward the XFCC header when included in the request.
◦ The Load Balancer is configured to pass through the TLS handshake via TCP to the instances of HAProxy, and ◦ HAProxy instance count is > 0	TLS terminated for the first time at HAProxy.	HAProxy sets the XFCC header with the client certificate received in the TLS handshake. The Gorouter forwards the header. **Breaking Change**: In the **Router behavior for Client Certificates** field, you cannot select the **Router does not request client certificates** option.
◦ The Load Balancer is configured to pass through the TLS handshake via TCP to instances of the Gorouter	TLS terminated for the first time at the Gorouter.	The Gorouter strips the XFCC header if it is included in the request and forwards the client certificate received in the TLS handshake in a new XFCC header. If you have deployed instances of HAProxy, app traffic bypasses those instances in this configuration. If you have also configured your load balancer to route requests for ssh directly to the Diego Brain, consider reducing HAProxy instances to 0. **Breaking Change**: In the **Router behavior for Client Certificates** field, you cannot select the **Router does not request client certificates** option.

For a description of the behavior of each configuration option, see Forward Client Certificate to Applications.

7. To configure Gorouter behavior for handling client certificates, select one of the following options in the **Router behavior for Client Certificate Validation** field.



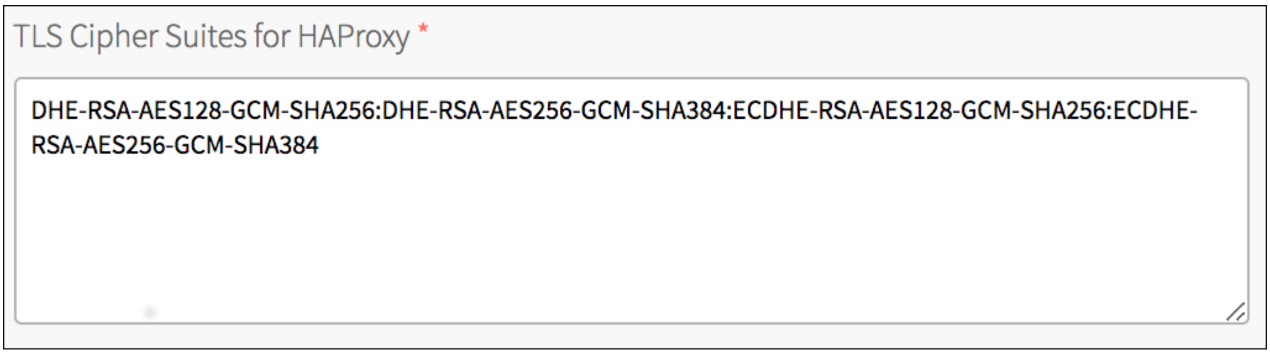
- **Router does not request client certificates.** This option is incompatible with the XFCC configuration options **TLS terminated for the first time at HAProxy** and **TLS terminated for the first time at the Router** in PAS because these options require mutual authentication. As client certificates are not requested, client will not provide them, and thus validation of client certificates will not occur.
- **Router requests but does not require client certificates.** Gorouter requests client certificates in TLS handshakes, validates them when presented, but does not require them. This is the default configuration.
- **Router requires client certificates.** Gorouter validates that the client certificate is signed by a Certificate Authority that Gorouter trusts. If Gorouter cannot validate the client certificate, the TLS handshake fails.

8. In the **TLS Cipher Suites for Router** field, specify the TLS cipher suites to use for TLS handshakes between the Gorouter and downstream clients like load balancers or HAProxy. Use an ordered, colon-delimited list of Golang-supported TLS cipher suites in the OpenSSL format. The recommended setting is `ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384`. Operators should verify that the ciphers are supported by any clients or downstream components that will initiate TLS handshakes with the Gorouter. For a list of TLS ciphers supported by the Gorouter, see Securing Traffic into Cloud Foundry.

Verify that whatever client is participating in the TLS handshake with the Gorouter has at least one cipher suite in common with the Gorouter.

> **Note**: Specify cipher suites that are supported by the versions configured in the **Minimum version of TLS supported by HAProxy and Router** field.

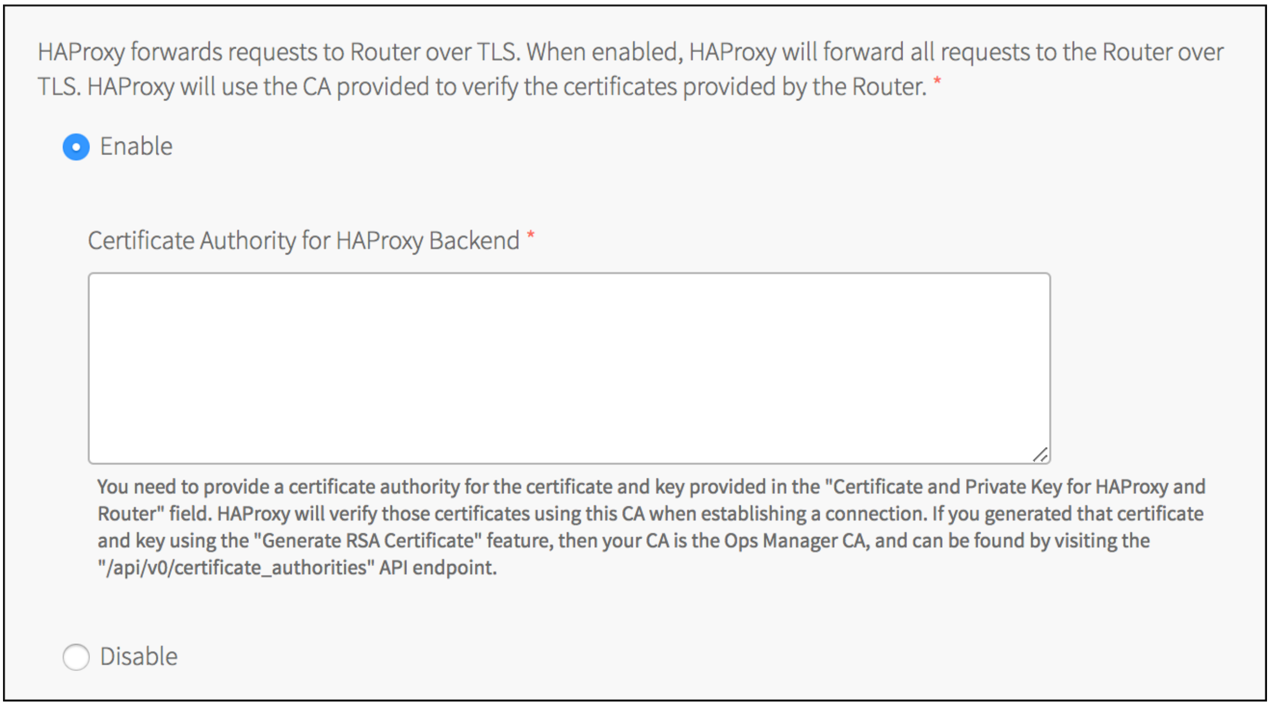
9. In the **TLS Cipher Suites for HAProxy** field, specify the TLS cipher suites to use in HAProxy for TLS handshakes between HAProxy and its clients such as load balancers and Gorouter. Use an ordered, colon-delimited list of TLS cipher suites in the OpenSSL format. The recommended setting: `DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384`
Operators should verify that the ciphers are supported by any clients or downstream components that will initiate TLS handshakes with the HAProxy.



Verify that whatever clients are participating in the TLS handshake with HAProxy have at least one cipher suite in common with HAProxy.

> **Note**: Specify cipher suites that are supported by the versions configured in the **Minimum version of TLS supported by HAProxy and Router** field.

10. Under **HAProxy forwards requests to Router over TLS**, select **Enable** or **Disable** based on your deployment layout.



- **Enable HAProxy forwarding of requests to Router over TLS**

If you want to:	Encrypt communication between HAProxy and the Gorouter
Then configure the following:	1. Leave **Enable** selected. 2. In the **Certificate Authority for HAProxy Backend** field, specify the Certificate Authority (CA) that signed the certificate you configured in the **Certificate and Private Key for HAProxy and Router** field. **Note**: If you used the **Generate RSA Certificate** link to generate a self-signed certificate, then the CA to specify is the Ops Manager CA, which you can locate at the CA endpoint in the Ops Manager API. 3. Make sure that Gorouter and HAPRoxy have TLS cipher suites in common in the **TLS Cipher Suites for Router** and **TLS Cipher Suites for HAProxy** fields.
See also:	◦ Terminating SSL/TLS at the Load Balancer and Gorouter ◦ Providing a Certificate for Your SSL/TLS Termination Point ◦ Using the Ops Manager API

- **Disable HAProxy forwarding of requests to Router over TLS**

If you want to:	Use non-encrypted communication between HAProxy and Gorouter, or you are not using HAProxy
Then configure the following:	1. Select **Disable**. 2. If you are not using HAProxy, set the number of HAProxy job instances to `0` on the **Resource Config** page. See Disable Unused Resources.
See also:	◦ Terminating SSL/TLS at the Gorouter Only ◦ Terminating SSL/TLS at the Load Balancer Only

11. If you are not using SSL encryption or if you are using self-signed certificates, select **Disable SSL certificate verification for this environment**. Selecting this checkbox also disables SSL verification for route services.

> **Note**: For production deployments, Pivotal does not recommend disabling SSL certificate verification.

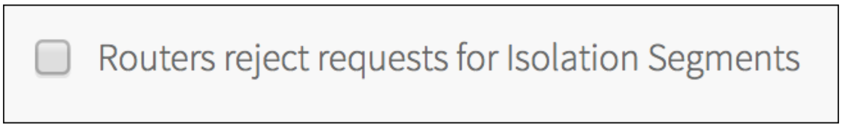
12. (Optional) If you want HAProxy or the Gorouter to reject any HTTP (non-encrypted) traffic, select the **Disable HTTP on HAProxy and Gorouter** checkbox. When selected, HAProxy and Gorouter will not listen on port 80.

13. (Optional) Select the **Disable insecure cookies on the Router** checkbox to set the secure flag for cookies generated by the router.

14. (Optional) To disable the addition of Zipkin tracing headers on the Gorouter, deselect the **Enable Zipkin tracing headers on the router** checkbox. Zipkin tracing headers are enabled by default. For more information about using Zipkin trace logging headers, see Zipkin Tracing in HTTP Headers.

15. (Optional) To stop the Router from writing access logs to local disk, deselect the **Enable Router to write access logs locally** checkbox. You should consider disabling this checkbox for high traffic deployments since logs may not be rotated fast enough and can fill up the disk.

16. By default, the PAS routers handle traffic for applications deployed to an isolation segment created by the PCF Isolation Segment tile. To configure the PAS routers to reject requests for applications within isolation segments, select the **Routers reject requests for Isolation Segments** checkbox.

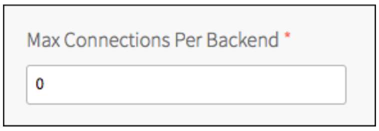
    Do not enable this option without deploying routers for each isolation segment. See the following topics for more information:
    - Installing PCF Isolation Segment
    - Sharding Routers for Isolation Segments.

17. In the **Choose whether or not to enable route services** section, choose either **Enable route services** or **Disable route services**. Route services are a class of marketplace services that perform filtering or content transformation on application requests and responses. See the Route Services topic for details.

18. (Optional) If you want to limit the number of app connections to the backend, enter a value in the **Max Connections Per Backend** field. You can use this field to prevent a poorly behaving app from all the connections and impacting other apps.

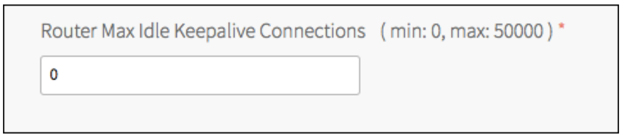
    To choose a value for this field, review the peak concurrent connections received by instances of the most popular apps in your deployment. You can determine the number of concurrent connections for an app from the `httpStartStop` event metrics emitted for each app request.

    If your deployment uses PCF Metrics, you can also obtain this peak concurrent connection information from Network Metrics. The default value is

    `500`.

19. Enter a value for **Router Max Idle Keepalive Connections**. See Considerations for Configuring max_idle_connections.

20. (Optional) To accommodate larger uploads over connections with high latency, increase the number of seconds in the **Router Timeout to Backends** field.

21. (Optional) Use the **Frontend Idle Timeout for Gorouter and HAProxy** field to help prevent connections from your load balancer to Gorouter or HAProxy from being closed prematurely. The value you enter sets the duration, in seconds, that Gorouter or HAProxy maintains an idle open connection from a load balancer that supports keep-alive.

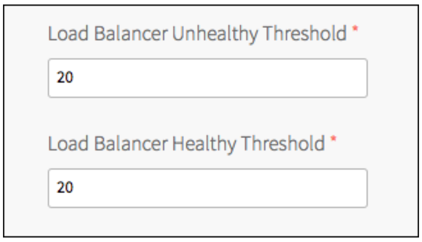
    In general, set the value higher than your load balancer's backend idle timeout to avoid the race condition where the load balancer sends a request before it discovers that Gorouter or HAProxy has closed the connection.

    See the following table for specific guidance and exceptions to this rule:

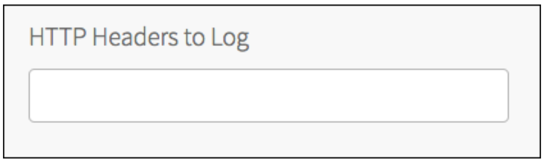
IaaS	Guidance
AWS	AWS ELB has a default timeout of 60 seconds, so Pivotal recommends a value greater than `60`.
Azure	By default, Azure load balancer times out at 240 seconds without sending a TCP RST to clients, so as an exception, Pivotal recommends a value lower than `240` to force the load balancer to send the TCP RST.
GCP	GCP has a default timeout of 600 seconds, so Pivotal recommends a value greater than `600`.
Other	Set the timeout value to be greater than that of the load balancer's backend idle timeout.

22. (Optional) Increase the value of **Load Balancer Unhealthy Threshold** to specify the amount of time, in seconds, that the router continues to accept connections before shutting down. During this period, healthchecks may report the router as unhealthy, which causes load balancers to failover to other routers. Set this value to an amount greater than or equal to the maximum time it takes your load balancer to consider a router instance unhealthy, given contiguous failed healthchecks.

23. (Optional) Modify the value of **Load Balancer Healthy Threshold**. This field specifies the amount of time, in seconds, to wait until declaring the Router instance started. This allows an external load balancer time to register the Router instance as healthy.

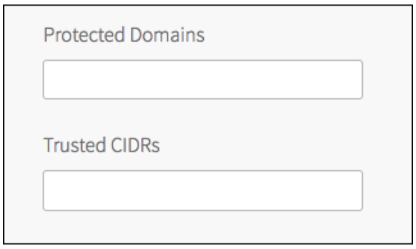


24. (Optional) If app developers in your organization want certain HTTP headers to appear in their app logs with information from the Gorouter, specify them in the **HTTP Headers to Log** field. For example, to support app developers that deploy Spring apps to PCF, you can enter Spring-specific HTTP headers.

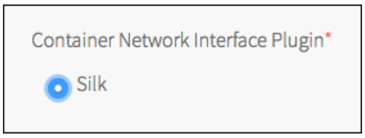


25. If you expect requests larger than the default maximum of 16 Kbytes, enter a new value (in bytes) for **HAProxy Request Max Buffer Size**. You may need to do this, for example, to support apps that embed a large cookie or query string values in headers.

26. If your PCF deployment uses HAProxy and you want it to receive traffic only from specific sources, use the following fields:
    - **Protected Domains:** Enter a comma-separated list of domains from which PCF can receive traffic.
    - **Trusted CIDRs:** Optionally, enter a space-separated list of CIDRs to limit which IP addresses from the **Protected Domains** can send traffic to PCF.



27. The **Loggregator Port** defaults to `443` if left blank. Enter a new value to override the default.



28. For **Container Network Plugin Interface**, ensure **Silk** is selected and review the following fields:

> **Note**: The **External** option exists to support NSX-T integration for vSphere deployments.

a. (Optional) You can change the value in the **Applications Network Maximum Transmission Unit (MTU)** field. Pivotal recommends setting the MTU value for your application network to `1454`. Some configurations, such as networks that use GRE tunnels, may require a smaller MTU value.

b. (Optional) Enter an IP range for the overlay network in the **Overlay Subnet** box. If you do not set a custom range, Ops Manager uses `10.255.0.0/16`.

> **WARNING**: The overlay network IP range must not conflict with any other IP addresses in your network.

c. Enter a UDP port number in the **VXLAN Tunnel Endpoint Port** box. If you do not set a custom port, Ops Manager uses 4789.

d. For **Denied logging interval**, set the per-second rate limit for packets blocked by either a container-specific networking policy or by

[Application Security Group](#) rules applied across the space, org, or deployment. This field defaults to `1`.

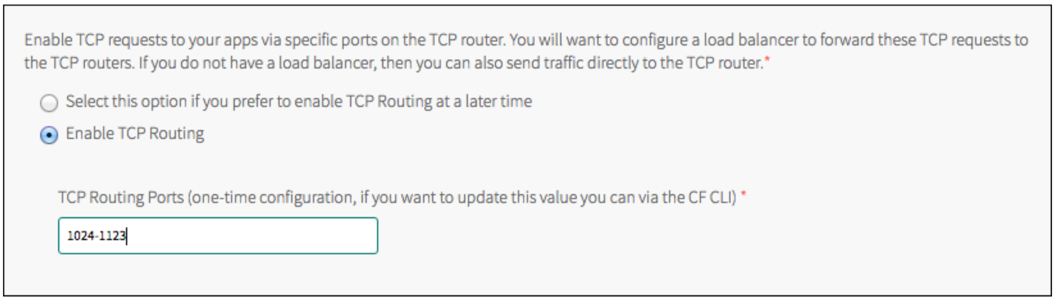
e. For **UDP logging interval**, set the per-second rate limit for UDP packets sent and received. This field defaults to `100`.
f. To enable logging for app traffic, select **Log traffic for all accepted/denied application packets**. See [Manage Logging for Container-to-Container Networking](#) for more information.

29. (Optional) TCP Routing is disabled by default. To enable this feature, perform the following steps:

    a. Select **Enable TCP Routing**.
    b. In **TCP Routing Ports**, enter a range of ports to be allocated for TCP Routes.

    For each TCP route you want to support, you must reserve a range of ports. This is the same range of ports you configured your load balancer with in the [Pre-Deployment Steps](#), unless you configured DNS to resolve the TCP domain name to the TCP router directly.

    The **TCP Routing Ports** field accepts a comma-delimited list of individual ports and ranges, for example `1024-1099,30000,60000-60099`. Configuration of this field is only applied on the first deploy, and update updates to the port range are made using the cf CLI. For details about modifying the port range, see the [Router Groups](#) topic.



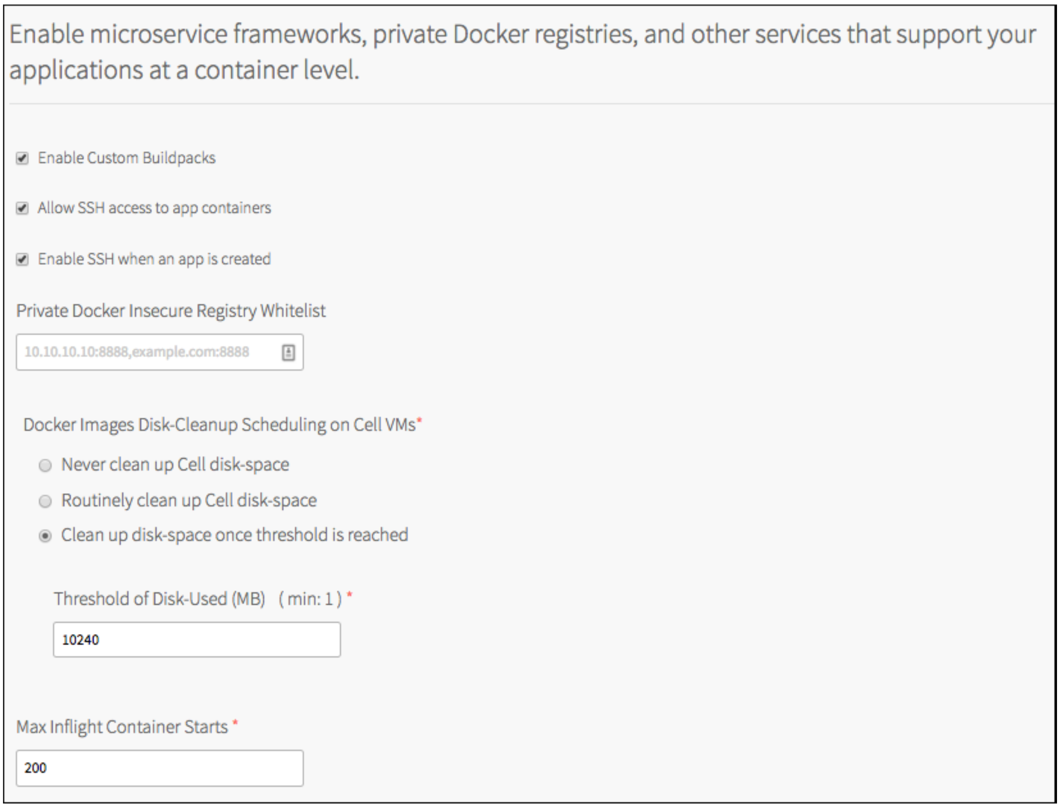
    c. For GCP, you also need to specify the name of a GCP TCP load balancer in the **LOAD BALANCER** column of TCP Router job of the **Resource Config** screen. You configure this later on in PAS. See [Configure Load Balancers](#) section of this topic.

30. (Optional) To disable TCP routing, click **Select this option if you prefer to enable TCP Routing at a later time** For more information, see the [Configuring TCP Routing in PAS](#) topic.

31. Click **Save**.

Step 7: Configure Application Containers

1. Select **Application Containers**.



2. The **Enable Custom Buildpacks** checkbox governs the ability to pass a custom buildpack URL to the `-b` option of the `cf push` command. By default, this ability is enabled, letting developers use custom buildpacks when deploying apps. Disable this option by disabling the checkbox. For more information about custom buildpacks, refer to the buildpacks section of the PCF documentation.

3. The **Allow SSH access to app containers** checkbox controls SSH access to application instances. Enable the checkbox to permit SSH access across your deployment, and disable it to prevent all SSH access. See the Application SSH Overview topic for information about SSH access permissions at the space and app scope.

4. If you want enable SSH access for new apps by default in spaces that allow SSH, select **Enable SSH when an app is created**. If you deselect the checkbox, developers can still enable SSH after pushing their apps by running `cf enable-ssh APP-NAME`.

5. You can configure Pivotal Application Service (PAS) to run app instances in Docker containers by supplying their IP address range(s) in the **Private Docker Insecure Registry Whitelist** textbox. See the Using Docker Registries topic for more information.

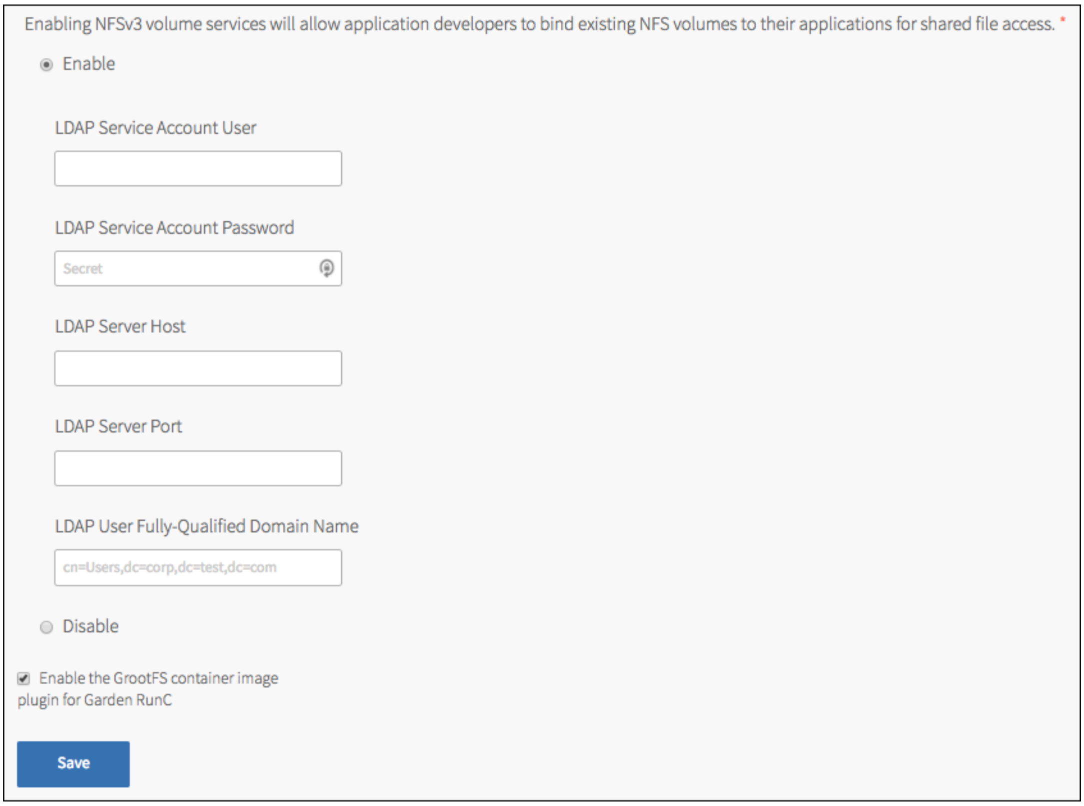
6. Select your preference for **Docker Images Disk-Cleanup Scheduling on Cell VMs**. If you choose **Clean up disk-space once threshold is reached**, enter a **Threshold of Disk-Used** in megabytes. For more information about the configuration options and how to configure a threshold, see Configuring Docker Images Disk-Cleanup Scheduling.

7. Enter a number in the **Max Inflight Container Starts** textbox. This number configures the maximum number of started instances across your deployment's Diego Cells. For more information about this feature, see Setting a Maximum Number of Started Containers.

8. Under **Enabling NFSv3 volume services**, select **Enable** or **Disable**. NFS volume services allow application developers to bind existing NFS volumes to their applications for shared file access. For more information, see the Enabling NFS Volume Services topic.

   > **Note**: In a clean install, NFSv3 volume services is enabled by default. In an upgrade, NFSv3 volume services is set to the same setting as it was in the previous deployment.

9. (Optional) To configure LDAP for NFSv3 volume services, perform the following steps:



- For **LDAP Service Account User**, enter the username of the service account in LDAP that will manage volume services.
- For **LDAP Service Account Password**, enter the password for the service account.
- For **LDAP Server Host**, enter the hostname or IP address of the LDAP server.
- For **LDAP Server Port**, enter the LDAP server port number. If you do not specify a port number, Ops Manager uses 389.
- For **LDAP Server Protocol**, enter the server protocol. If you do not specify a protocol, Ops Manager uses TCP.
- For **LDAP User Fully-Qualified Domain Name**, enter the fully qualified path to the LDAP service account. For example, if you have a service account named `volume-services` that belongs to organizational units (OU) named `service-accounts` and `my-company`, and your domain is named `domain`, the fully qualified path looks like the following:

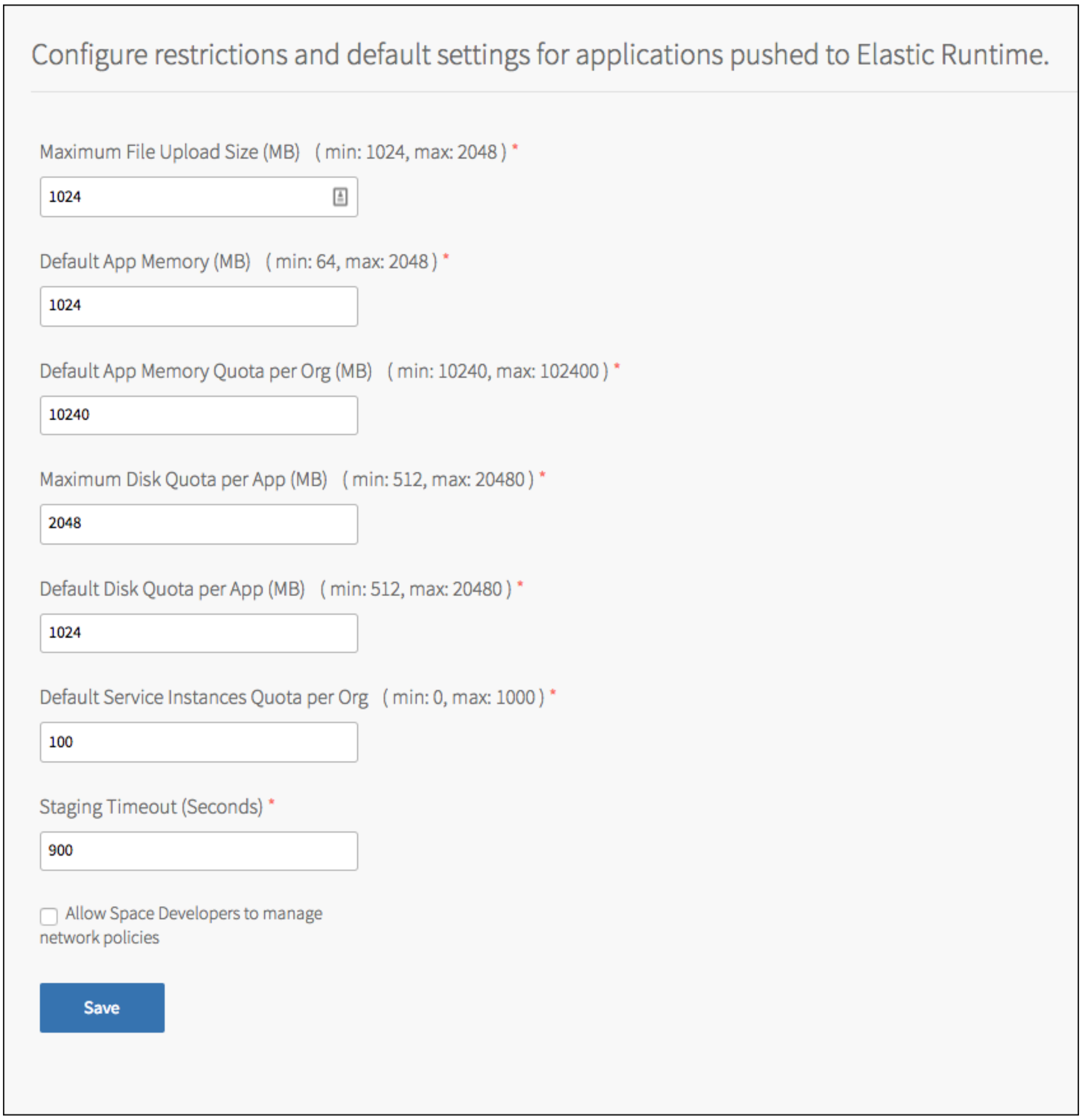
  ```
  CN=volume-services,OU=service-accounts,OU=my-company,DC=domain,DC=com
  ```

10. By default, PAS manages container images using the GrootFS plugin for Garden-runC. If you experience issues with GrootFS, you can disable the plugin and use the image plugin built into Garden-runC.

11. Click **Save**.

Step 8: Configure Application Developer Controls

1. Select **Application Developer Controls**.



2. Enter the **Maximum File Upload Size (MB)**. This is the maximum size of an application upload.

3. Enter the **Default App Memory (MB)**. This is the amount of RAM allocated by default to a newly pushed application if no value is specified with the cf CLI.

4. Enter the **Default App Memory Quota per Org**. This is the default memory limit for all applications in an org. The specified limit only applies to the first installation of PAS. After the initial installation, operators can use the cf CLI to change the default value.

5. Enter the **Maximum Disk Quota per App (MB)**. This is the maximum amount of disk allowed per application.

> **Note**: If you allow developers to push large applications, PAS may have trouble placing them on Cells. Additionally, in the event of a system upgrade or an outage that causes a rolling deploy, larger applications may not successfully re-deploy if there is insufficient disk capacity. Monitor your deployment to ensure your Cells have sufficient disk to run your applications.

6. Enter the **Default Disk Quota per App (MB)**. This is the amount of disk allocated by default to a newly pushed application if no value is specified with the cf CLI.

7. Enter the **Default Service Instances Quota per Org**. The specified limit only applies to the first installation of PAS. After the initial installation, operators can use the cf CLI to change the default value .

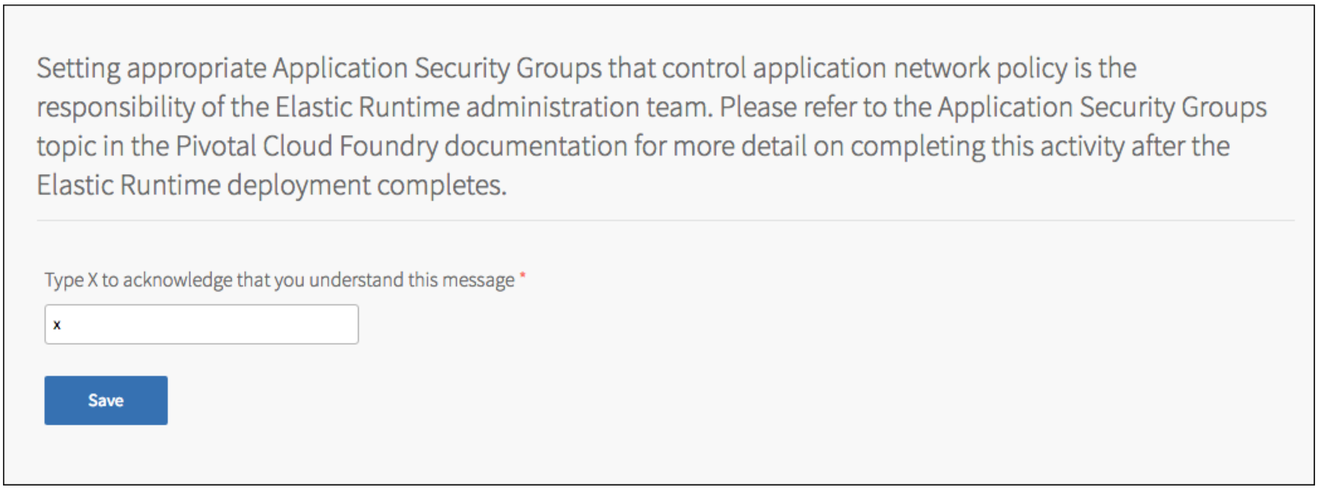
8. Enter the **Staging Timeout (Seconds)**. When you stage an application droplet with the Cloud Controller, the server times out after the number of seconds you specify in this field.

9. Select the **Allow Space Developers to manage network policies** checkbox to permit developers to manage their own network policies for their applications.

10. Click **Save**.

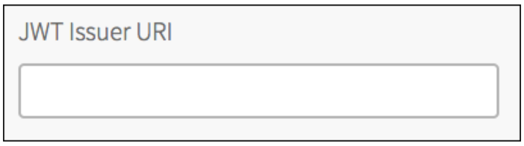
Step 9: Review Application Security Groups

Setting appropriate Application Security Groups is critical for a secure deployment. Type `X` in the box to acknowledge that once the Pivotal Application Service (PAS) deployment completes, you will review and set the appropriate application security groups. See Restricting App Access to Internal PCF Components for instructions.



Step 10: Configure UAA

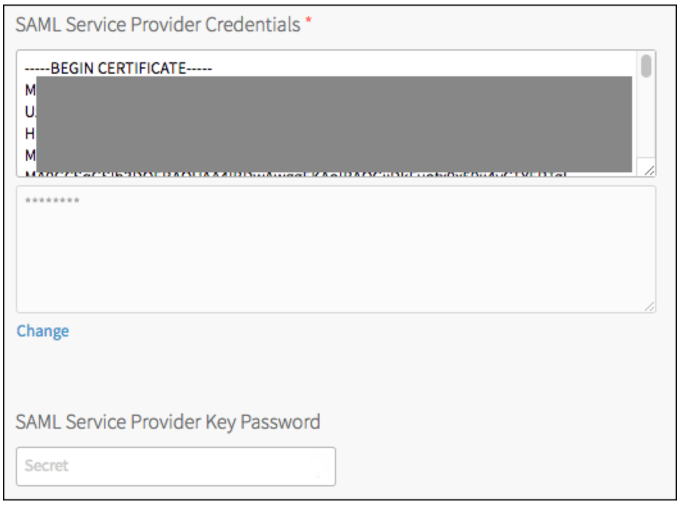
1. Select **UAA**.
2. (Optional) Under **JWT Issuer URI**, enter the URI that UAA uses as the issuer when generating tokens.



3. Under **SAML Service Provider Credentials**, enter a certificate and private key to be used by UAA as a SAML Service Provider for signing outgoing SAML authentication requests. You can provide an existing certificate and private key from your trusted Certificate Authority or generate a self-signed certificate. The following domain must be associated with the certificate: `*.login.YOUR-SYSTEM-DOMAIN`.

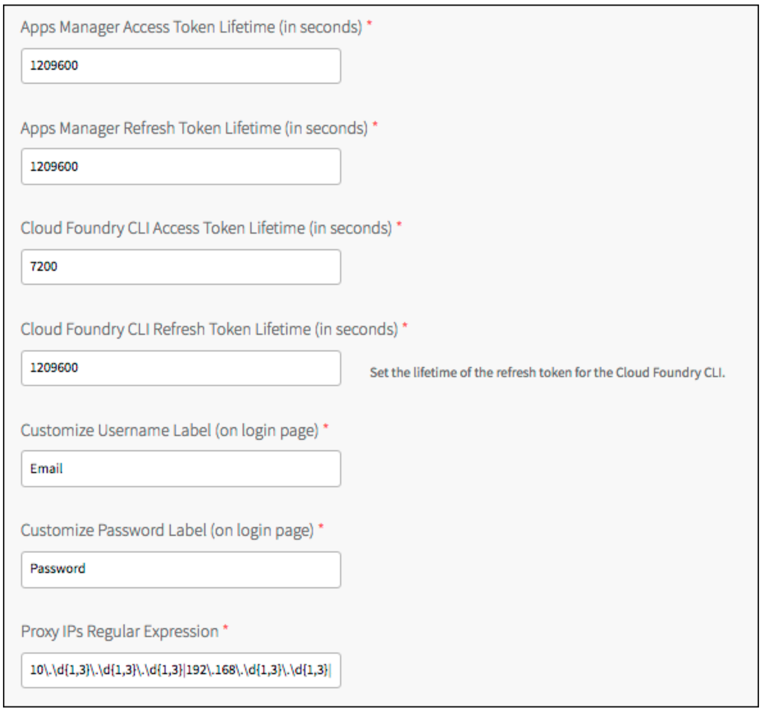
Note: The Pivotal Single Sign-On Service and Pivotal Spring Cloud Services tiles require the `*.login.YOUR-SYSTEM-DOMAIN`.

4. If the private key specified under **Service Provider Credentials** is password-protected, enter the password under **SAML Service Provider Key Password**.



5. For **Signature Algorithm**, choose an algorithm from the dropdown menu to use for signed requests and assertions. The default value is `SHA256`.

6. (Optional) In the **Apps Manager Access Token Lifetime**, **Apps Manager Refresh Token Lifetime**, **Cloud Foundry CLI Access Token Lifetime**, and **Cloud Foundry CLI Refresh Token Lifetime** fields, change the lifetimes of tokens granted for Apps Manager and Cloud Foundry Command Line Interface (cf CLI) login access and refresh. Most deployments use the defaults.



7. (Optional) Customize the text prompts used for username and password from the cf CLI and Apps Manager login popup by entering values for **Customize Username Label (on login page)** and **Customize Password Label (on login page)**.

8. (Optional) The **Proxy IPs Regular Expression** field contains a pipe-delimited set of regular expressions that UAA considers to be reverse proxy IP addresses. UAA respects the `x-forwarded-for` and `x-forwarded-proto` headers coming from IP addresses that match these regular expressions. To configure UAA to respond properly to Gorouter or HAProxy requests coming from a public IP address, append a regular expression or regular expressions to match the public IP address.

9. You can configure UAA to use the internal MySQL database provided with PCF, or you can configure an external database provider. Follow the procedures in either the Internal Database Configuration or the External Database Configuration section below.

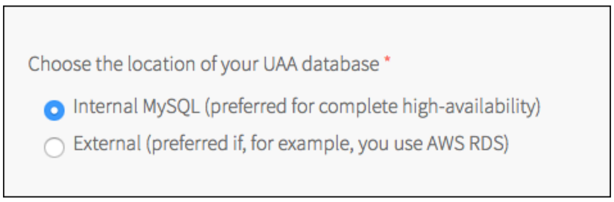
> **Note**: For GCP installations, Pivotal recommends selecting **External** and using Google Cloud SQL.

> **Note**: If you are performing an upgrade, do not modify your existing internal database configuration or you may lose data. You must migrate your existing data before changing the configuration. See Upgrading Pivotal Cloud Foundry for additional upgrade information, and contact Pivotal Support for help.

Internal Database Configuration

If you chose to not deploy a Google Cloud SQL database with Terraform, follow these steps.

1. Select **Internal MySQL**.



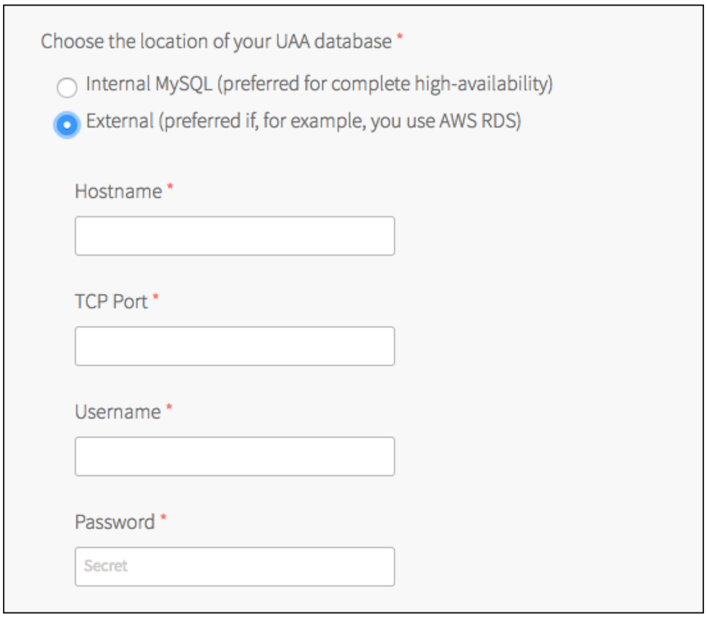
2. Click **Save**.

3. Ensure that you complete the “Configure Internal MySQL” step later in this topic to configure high availability and automatic backups for your internal MySQL databases.

External Database Configuration

If you chose to deploy a Google Cloud SQL database with Terraform, follow these steps.

1. From the **UAA** section in Pivotal Application Service (PAS), select **External**.

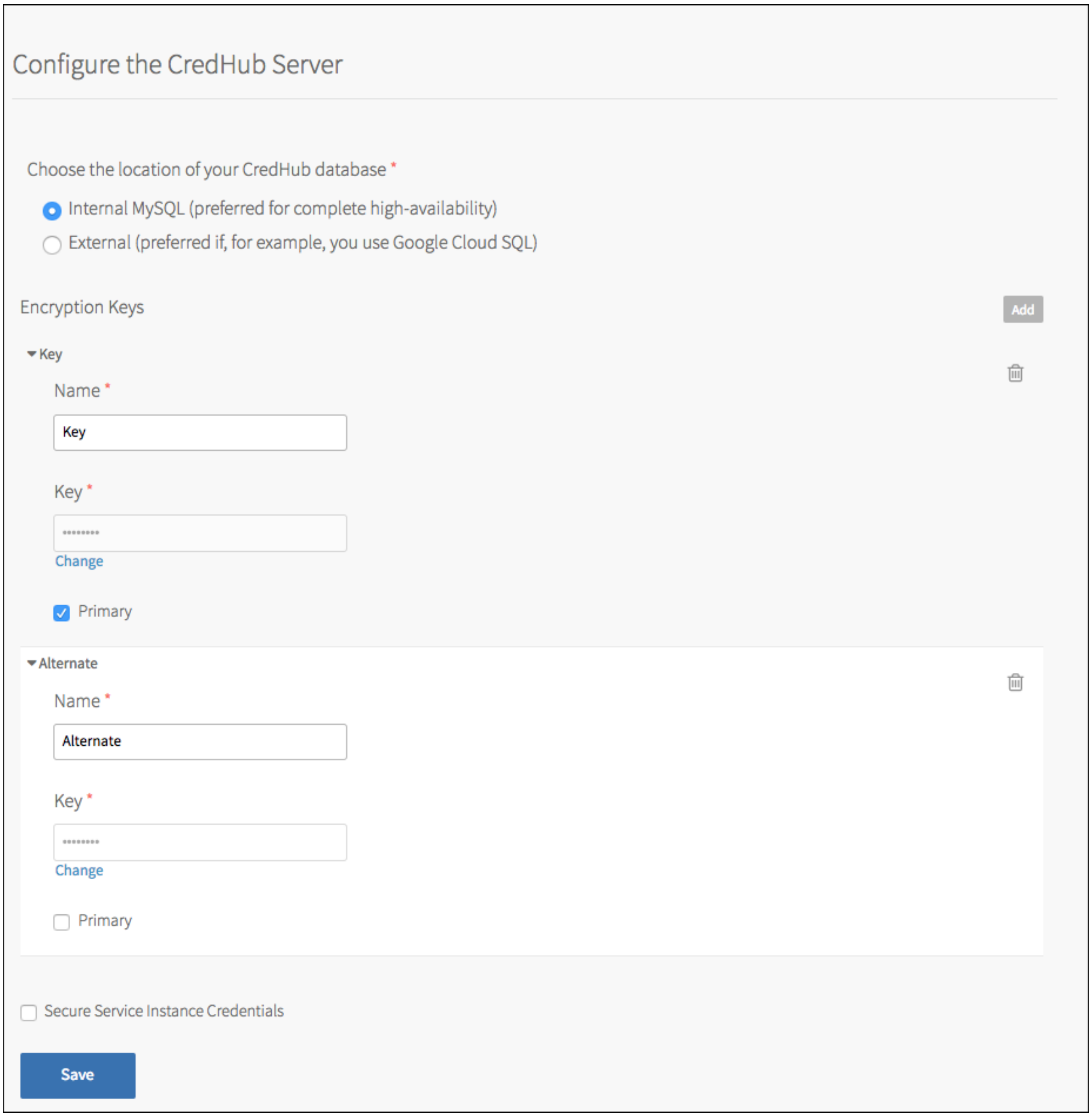


2. Complete the fields as follows:
   - **Hostname**: Enter the value of `sql_db_ip` from your Terraform output.
   - **TCP Port**: Enter `3306`.
   - **User Account and Authentication database username**: Enter the value of `pas_sql_username` from your Terraform output.
   - **User Account and Authentication database password**: Enter the value of `pas_sql_password` from your Terraform output.

3. Click **Save**.

Step 11: (Optional) Configure CredHub

1. Select **Credhub**.



2. Choose the location of your CredHub Database. PAS includes this CredHub database for services to store their service instance credentials.

   a. If you chose **External**, enter the following:
      - **Hostname**: The IP address of your database server. This is the value of `sql_db_ip` from your Terraform output.
      - **TCP Port**: The port of your database server. Enter `3306`.
      - **Username**: The value of `pas_sql_username` from your Terraform output.
      - **Password**: The value of `pas_sql_password` from your Terraform output.
      - **Database CA Certificate**: Enter a certificate to use for encrypting traffic to and from the database.

3. Under **Encryption Keys**, specify a key to use for encrypting and decrypting the values stored in the CredHub database.
   - **Name**: Enter the name of the key.
   - **Key**: Enter a key that is at least 20 characters in length.
   - **Primary**: Select this checkbox to use this key as your primary key.

   > **Note**: Ensure that you only mark one key as **Primary**. The UI includes an **Add** button to add more keys to support key rotation. For more information, see the Rotating PAS CredHub Encryption Keys topic.

4. If your deployment uses any PCF services that support storing service instance credentials in CredHub and you want to enable this feature, select the **Secure Service Instance Credentials** checkbox.

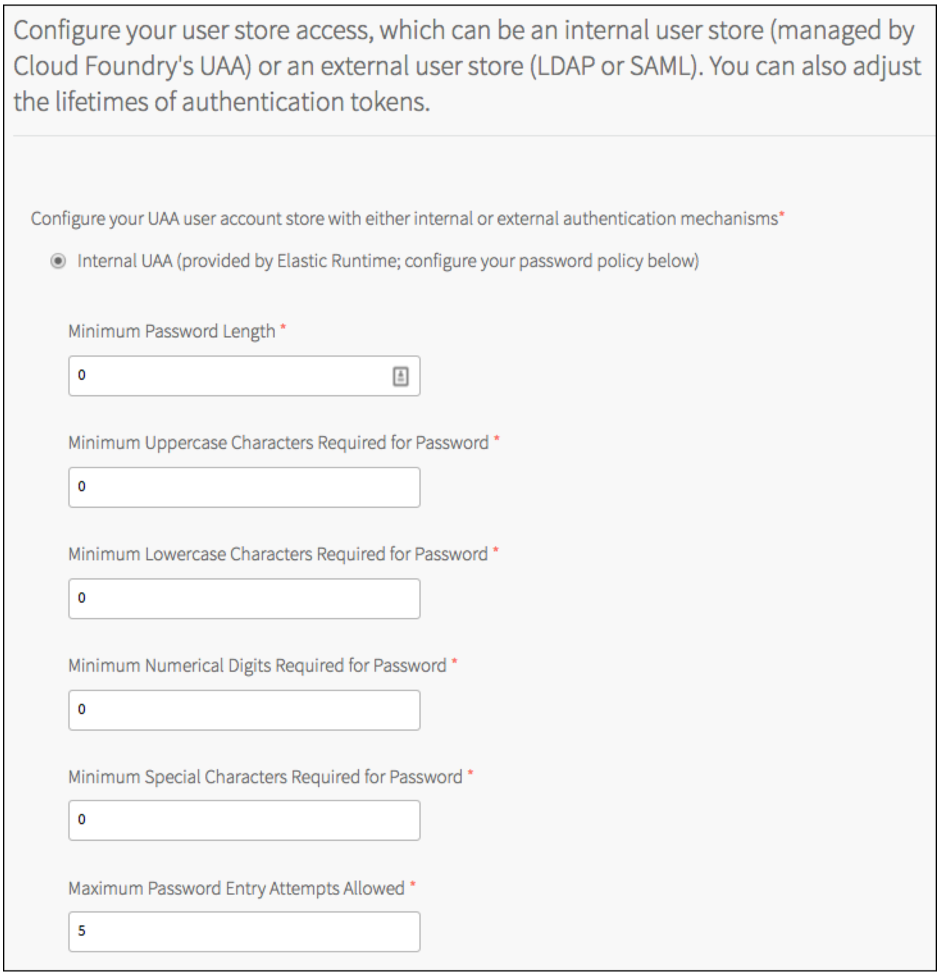
5. Select the **Resource Config** pane.

6. Under the **Job** column of the **CredHub** row, set the number of instances to `2`. This is the minimum instance count required for high availability.

Note: To use the Runtime CredHub feature, you must follow the additional steps in Securing Service Instance Credentials with Runtime CredHub.

Step 12: Configure Authentication and Enterprise SSO

1. Select **Authentication and Enterprise SSO**.



2. To authenticate user sign-ons, your deployment can use one of three types of user database: the UAA server's internal user store, an external SAML identity provider, or an external LDAP server.
   - To use the internal UAA, select the **Internal** option and follow the instructions in the Configuring UAA Password Policy topic to configure your password policy.
   - To connect to an external identity provider through SAML, scroll down to select the **SAML Identity Provider** option and follow the instructions in the Configuring PCF for SAML section of the *Configuring Authentication and Enterprise SSO for Pivotal Application Service (PAS)* topic.
   - To connect to an external LDAP server, scroll down to select the **LDAP Server** option and follow the instructions in the Configuring LDAP section of the *Configuring Authentication and Enterprise SSO for PAS* topic.
3. Click **Save**.

Step 13: Configure System Databases

You can configure PAS to use Google Cloud SQL for the databases required by PAS.

Note: If you are performing an upgrade, do not modify your existing internal database configuration or you may lose data. You must migrate your existing data first before changing the configuration. Contact Pivotal Support for help. See Upgrading Pivotal Cloud Foundry for additional upgrade information.

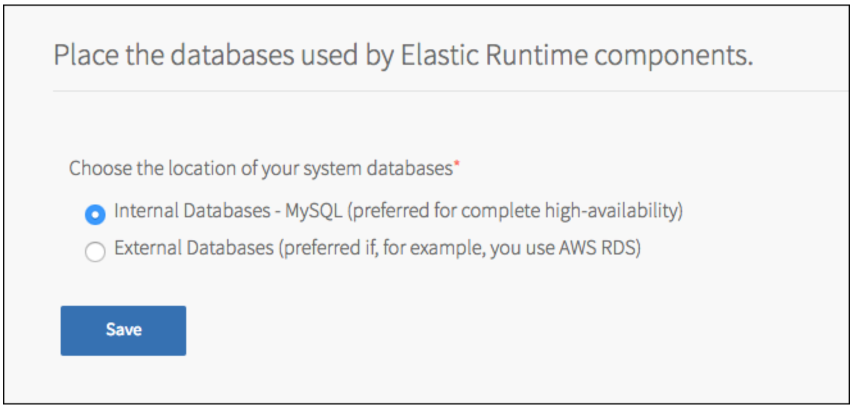
Internal Database Configuration

> **Note**: For GCP installations, Pivotal recommends selecting **External** and using Google Cloud SQL. Only use internal MySQL for non-production or test installations on GCP.

> **Note**: Follow these steps if you did not modify your Terraform variables file to deploy an Google CloudSQL instance.

If you want to use internal databases for your deployment, perform the following steps:

1. Select **Databases**.



2. Select **Internal Databases - MySQL**.
3. Click **Save**.

Then proceed to Step 14: (Optional) Configure Internal MySQL to configure high availability and automatic backups for your internal MySQL databases.

External Database Configuration

> **Note**: Follow these steps if you modified your Terraform variables file to deploy an Google CloudSQL instance.

Pivotal recommends using an external database such as Google Cloud SQL for high availability reasons.

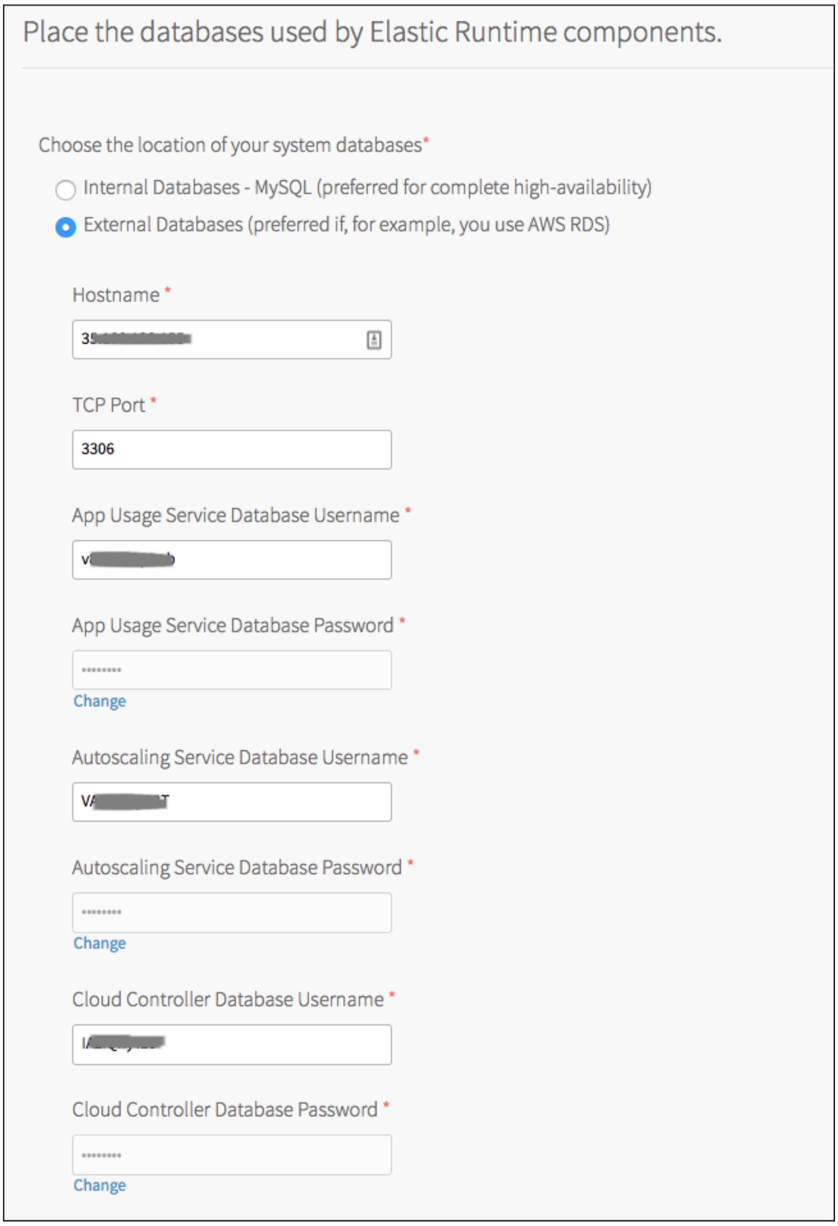
On GCP, you can use Google Cloud SQL and use the automated backup and high availability replica features.

> **Note**: To configure an external database for UAA, see the *External Database Configuration* section of Configure UAA.

> **Warning**: Protect whichever database you use in your deployment with a password.

To specify your PAS databases, perform the following steps:

1. Select the **External Databases** option.
2. Complete the following fields:
   - **Hostname**: Enter the value of `sql_db_ip` from your Terraform output.
   - **TCP Port**: Enter `3306`.
   - For the username and password field for each relational database, enter the values of `pas_sql_username` and `pas_sql_password` from your



Terraform output.

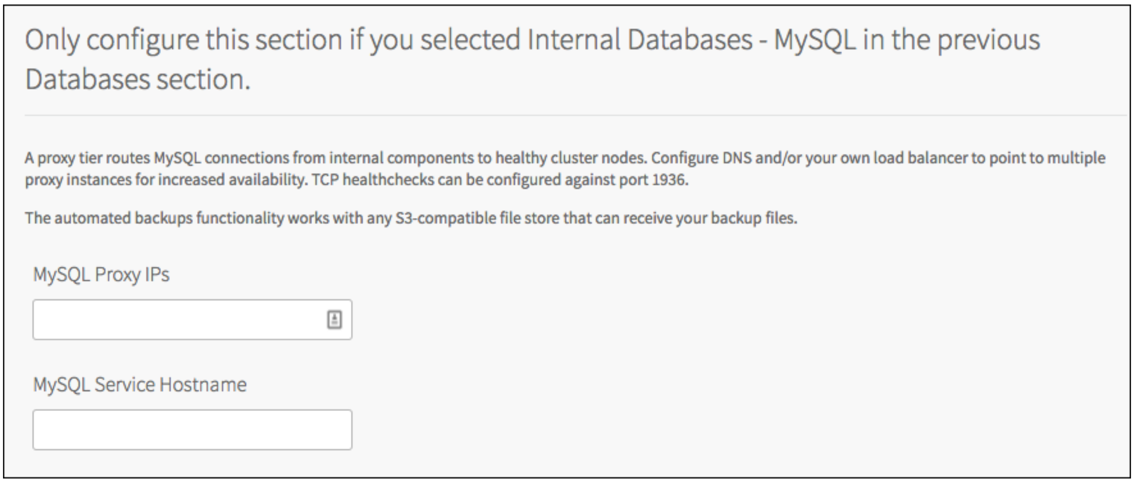
3. Click **Save**.

Step 14: (Optional) Configure Internal MySQL

Note: You only need to configure this section if you have selected **Internal Databases - MySQL** in the **Databases** section.

1. Select **Internal MySQL**.

2. In the **MySQL Proxy IPs** field, enter one or more comma-delimited IP addresses that are not in the reserved CIDR range of your network. If a MySQL node fails, these proxies re-route connections to a healthy node. See the MySQL Proxy topic for more information.



3. For **MySQL Service Hostname**, enter an IP address or hostname for your load balancer. If a MySQL proxy fails, the load balancer re-routes connections to a healthy proxy. If you leave this field blank, components are configured with the IP address of the first proxy instance entered above.

> **Warning:** You must configure a load balancer to achieve complete high-availability.

4. In the **Replication canary time period** field, leave the default of 30 seconds or modify the value based on the needs of your deployment. Lower numbers cause the canary to run more frequently, which means that the canary reacts more quickly to replication failure but adds load to the database.

5. In the **Replication canary read delay** field, leave the default of 20 seconds or modify the value based on the needs of your deployment. This field configures how long the canary waits, in seconds, before verifying that data is replicating across each MySQL node. Clusters under heavy load can experience a small replication lag as write-sets are committed across the nodes.

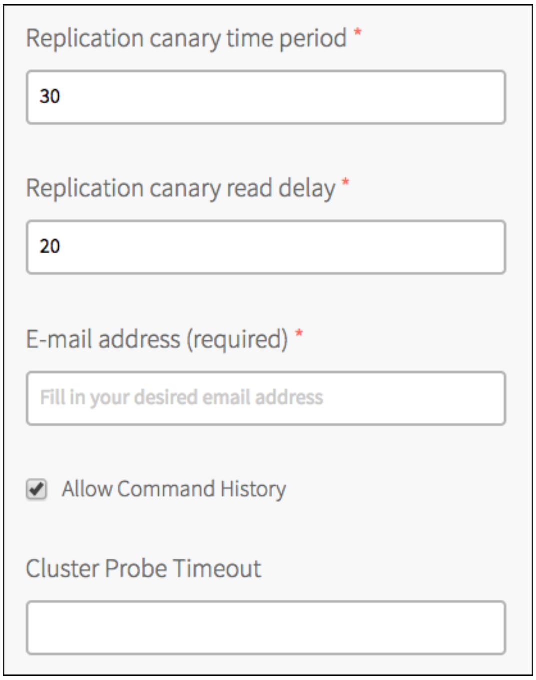
6. (**Required**): In the **E-mail address** field, enter the email address where the MySQL service sends alerts when the cluster experiences a replication issue or when a node is not allowed to auto-rejoin the cluster.

7. To prohibit the creation of command line history files on the MySQL nodes, disable the **Allow Command History** checkbox.

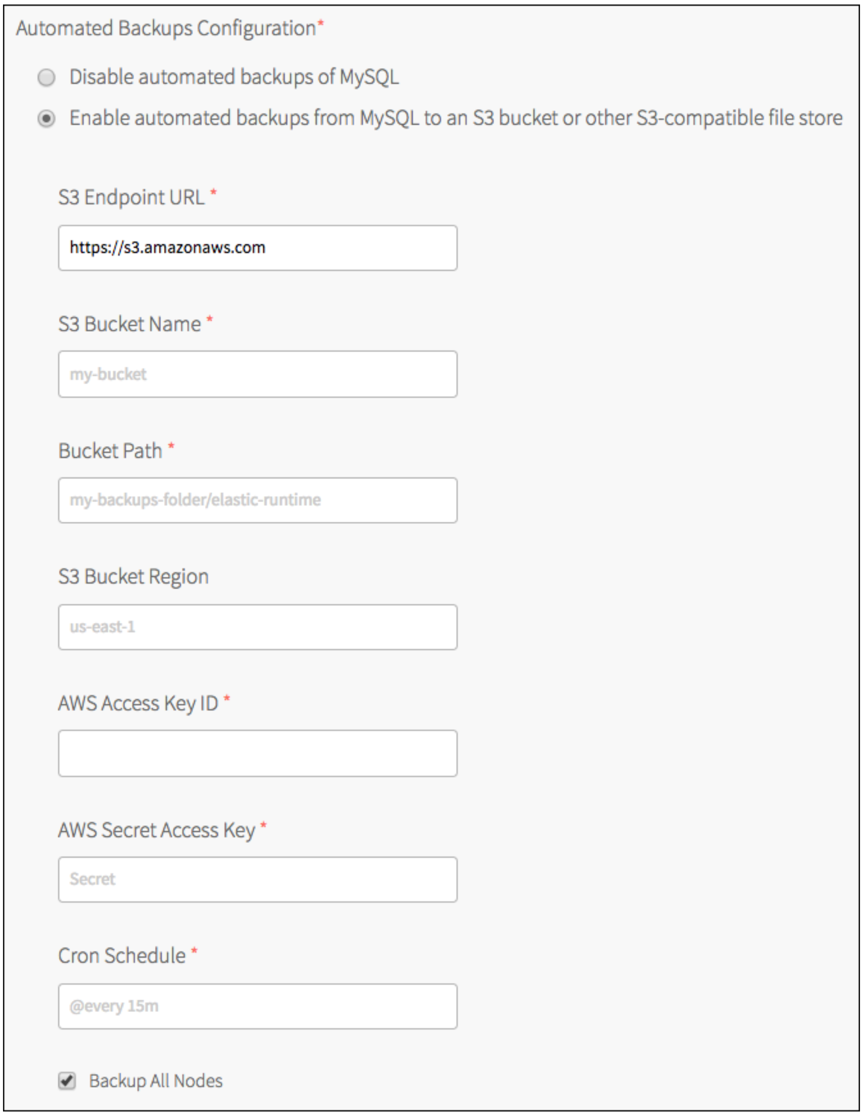
8. To allow the admin and roadmin to connect from any remote host, enable the **Allow Remote Admin Access** checkbox. When the checkbox is disabled, admins must `bosh ssh` into each MySQL VM to connect as the MySQL super user.

> **Note:** Network configuration and Application Security Groups restrictions may still limit a client's ability to establish a connection with the databases.

9. For **Cluster Probe Timeout**, enter the maximum amount of time, in seconds, that a new node will search for existing cluster nodes. If left blank, the default value is 10 seconds.



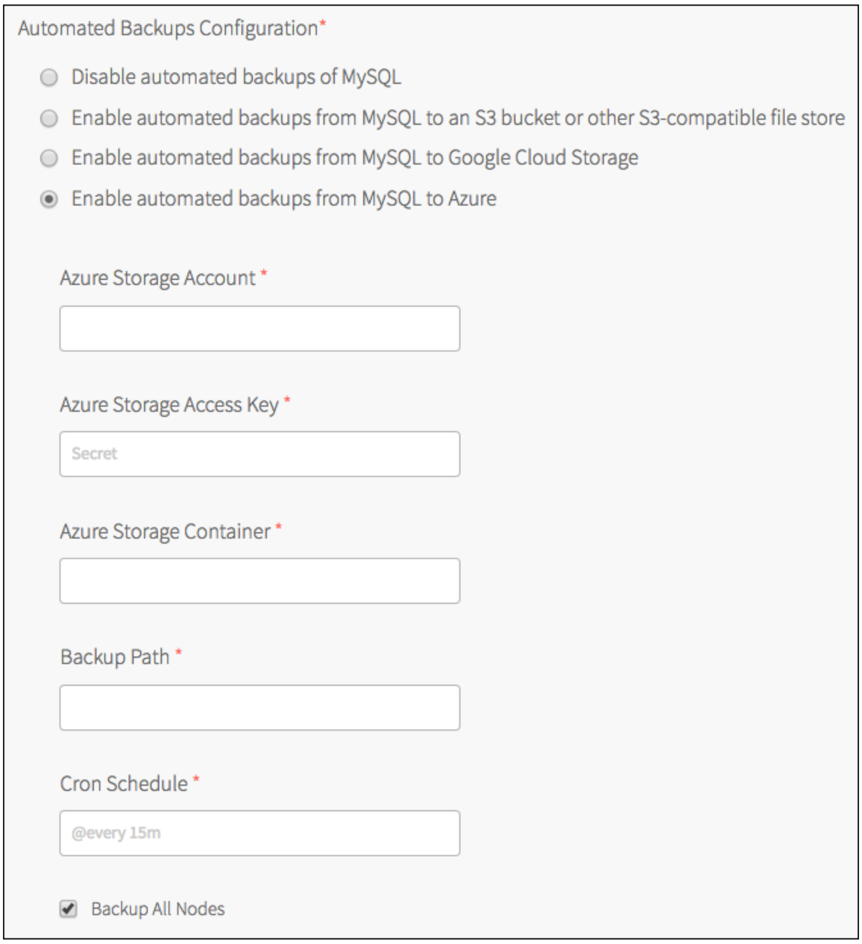
10. Under **Automated Backups Configuration**, choose one of three options for MySQL backups:
    - **Disable automatic backups of MySQL** disables automatic backups, but you can still deploy the Backup Prepare Node if you use BOSH Backup and Restore to back up your MySQL database. For more information, see the Backing Up Pivotal Cloud Foundry with BBR topic.
    - **Enable automated backups from MySQL to an S3 bucket or other S3-compatible file store** saves your backups to an existing Amazon Web Services (AWS) or Ceph S3-compatible blobstore.



This option requires the following fields:

- For **S3 Bucket Name**, enter the name of your S3 bucket. Do not include an `s3://` prefix, a trailing `/`, or underscores. If the bucket does not already exist, it will be created automatically.
- For **Bucket Path**, specify a folder within the bucket to hold your MySQL backups. Do not include a trailing `/`.
- For **S3 Bucket Region**, enter the AWS region where the bucket is located, such as `us-east-1`.
- For **AWS Access Key ID** and **AWS Secret Access Key**, enter your AWS or Ceph credentials.
- For **Cron Schedule**, enter a valid cron expression to schedule your automated backups. Cron uses your computer's local time zone.
- Enable **Backup All Nodes** to make unique backups from each instance of the MySQL server rather than just the first MySQL server instance.

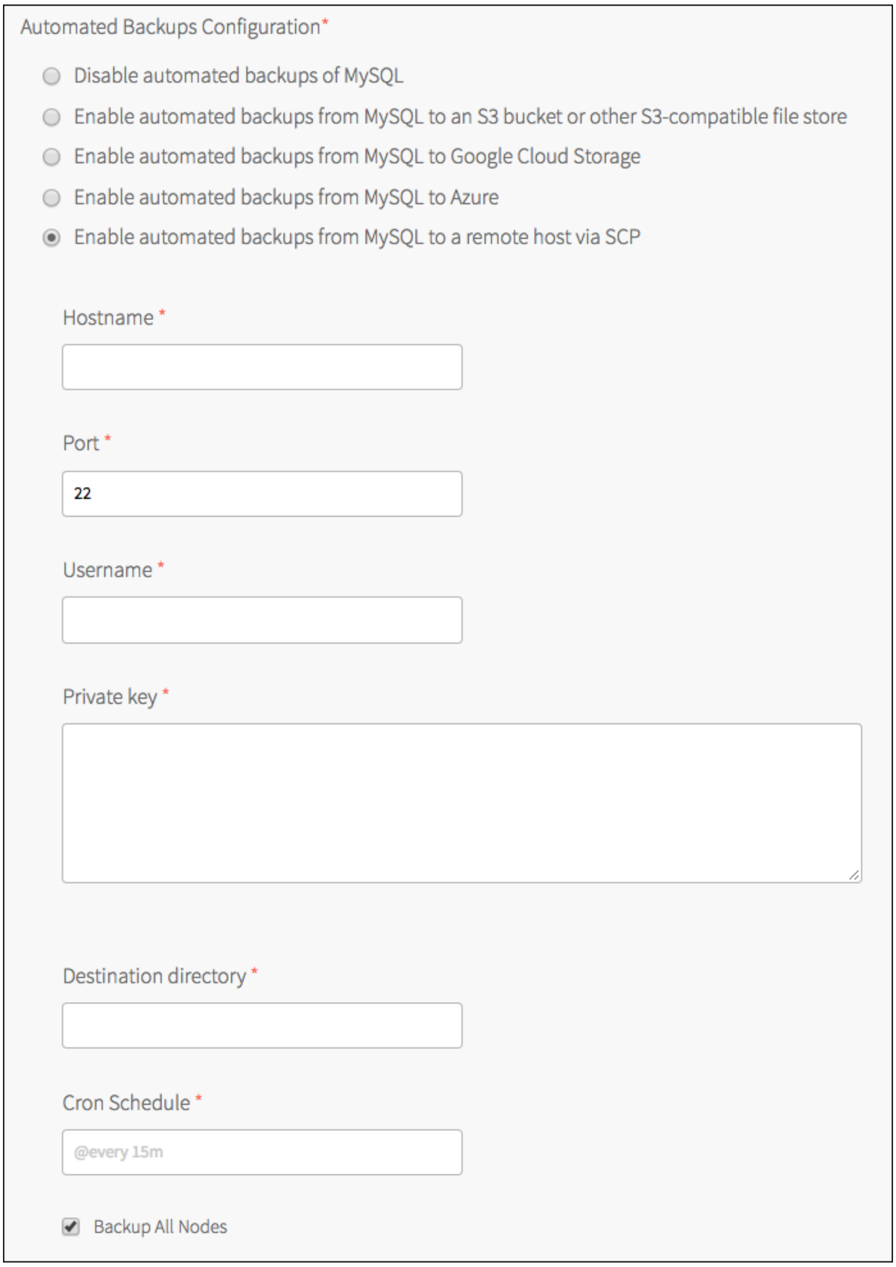
- **Enable automated backups from MySQL to Azure** saves your backups to Azure.



This option requires the following fields:

- For **Azure Storage Account**, enter the name of an existing Azure storage account where backups will be uploaded. For more information about creating and managing an Azure storage account, see the Azure documentation.
- For **Azure Storage Access Key**, enter an Azure storage access key for the storage account.
- For **Azure Storage Container**, enter the name of an existing Azure storage container that will store the backups.
- For **Backup Path**, enter the path within the Azure storage container where backups will be uploaded.
- For **Cron Schedule**, enter a valid cron expression to schedule your automated backups. Cron uses your computer's local time zone.
- Enable **Backup All Nodes** to make unique backups from each instance of the MySQL server rather than just the first MySQL server instance.

- **Enable automated backups from MySQL to a remote host via SCP**saves your backups to a remote host using secure copy protocol (SCP).



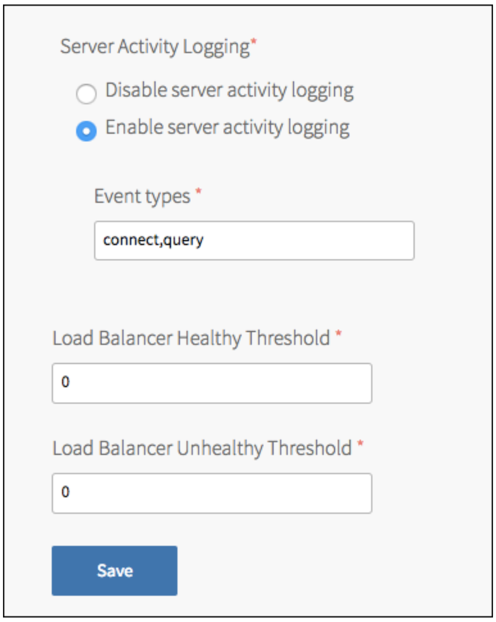
This option requires the following fields:

- For **Hostname**, enter the name of your SCP host.
- For **Port**, enter your SCP port. This should be the TCP port that your SCP host uses for SSH. The default port is `22`.
- For **Username**, enter your SSH username for the SCP host.
- For **Private key**, paste in your SSH private key.
- For **Destination directory**, enter the directory on the SCP host where you want to save backup files.
- For **Cron Schedule**, enter a valid cron expression to schedule your automated backups. Cron uses your computer's local time zone.
- Enable **Backup All Nodes** to make unique backups from each instance of the MySQL server rather than just the first MySQL server instance.

> **Note**: If you choose to enable automated MySQL backups, set the number of instances for the **Backup Prepare Node** under the **Resource Config** section of the Pivotal Application Service (PAS) tile to `1`.

11. If you want to log audit events for internal MySQL, select **Enable server activity logging** under **Server Activity Logging**.

    a. For the **Event types** field, you can enter the events you want the MySQL service to log. By default, this field includes `connect` and `query`, which tracks who connects to the system and what queries are processed. For more information, see the Logging Events section of the MariaDB documentation.



12. Enter values for the following fields:
    - **Load Balancer Healthy Threshold**: Specifies the amount of time, in seconds, to wait until declaring the MySQL proxy instance started. This allows an external load balancer time to register the instance as healthy.
    - **Load Balancer Unhealthy Threshold**: Specifies the amount of time, in seconds, that the MySQL proxy continues to accept connections before shutting down. During this period, the healthcheck reports as unhealthy to cause load balancers to fail over to other proxies. You must enter a value greater than or equal to the maximum time it takes your load balancer to consider a proxy instance unhealthy, given repeated failed healthchecks.

13. If you want to enable the MySQL interruptor feature, select the checkbox to **Prevent node auto re-join**. This feature stops all writes to the MySQL database if it notices an inconsistency in the dataset between the nodes. For more information, see the Interruptor section in the MySQL for PCF documentation.

14. Click **Save**.

Step 15: Configure File Storage

To minimize system downtime, Pivotal recommends using highly resilient and redundant*external* filestores for your Pivotal Application Service (PAS) file storage.

When configuring file storage for the Cloud Controller in PAS, you can select one of the following:

- Internal WebDAV filestore
- External S3-compatible or Ceph-compatible filestore
- External Google Cloud Storage

> **Note**: Use this option if you added `create_gcs_buckets = true` to your `terraform.tfvars` file.

- External Azure Cloud Storage

For production-level PCF deployments on GCP, Pivotal recommends selecting **External Google Cloud Storage**. For more information about production-level PCF deployments on GCP, see the Reference Architecture for Pivotal Cloud Foundry on GCP.

For additional factors to consider when selecting file storage, see the Considerations for Selecting File Storage in Pivotal Cloud Foundry topic.

Internal Filestore

Internal file storage is only appropriate for small, non-production deployments.

To use the PCF internal filestore, perform the following steps:

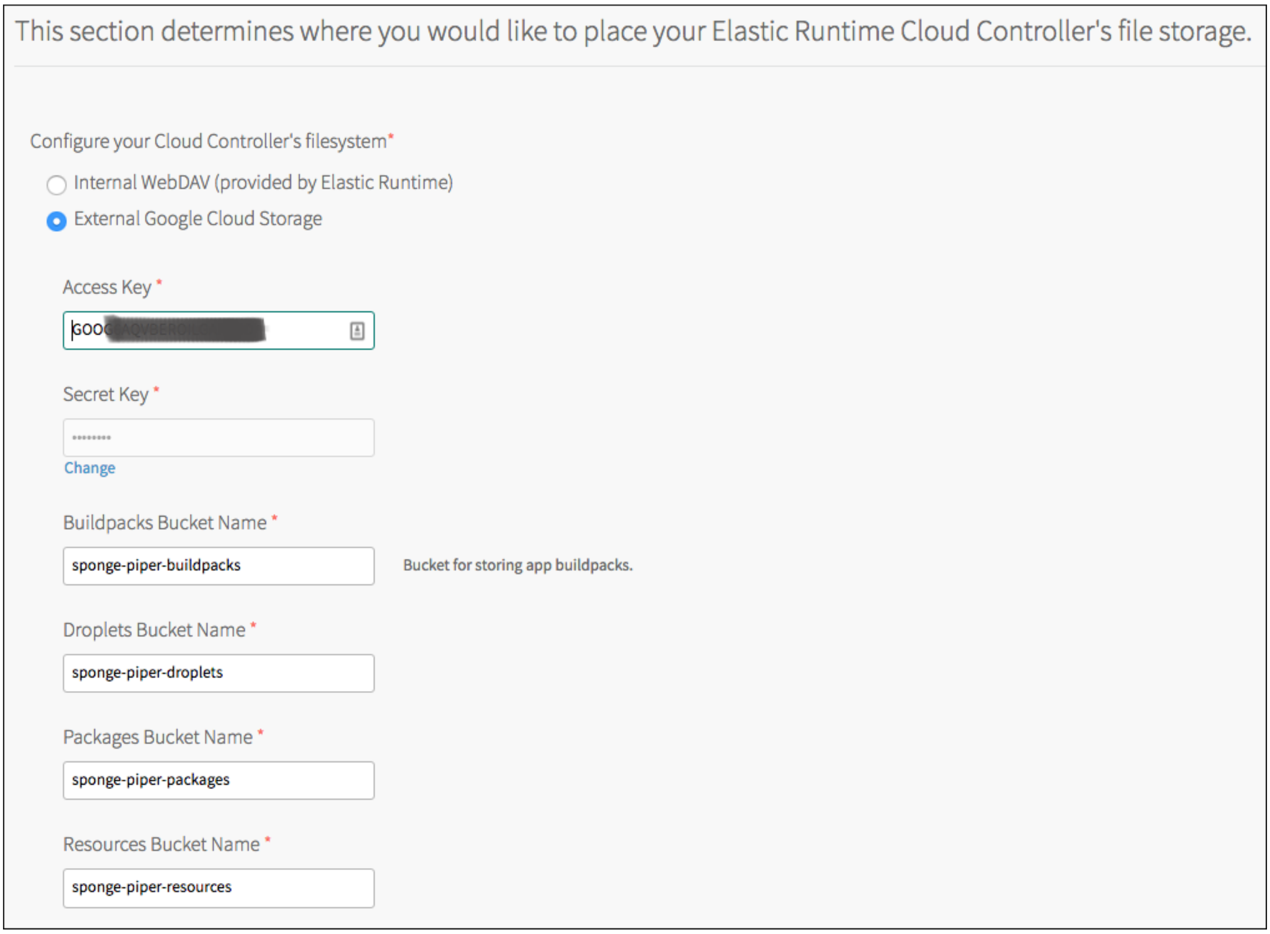
1. In the Pivotal Application Service (PAS) tile, select **File Storage**.

2. Select **Internal WebDAV**, and click **Save**.

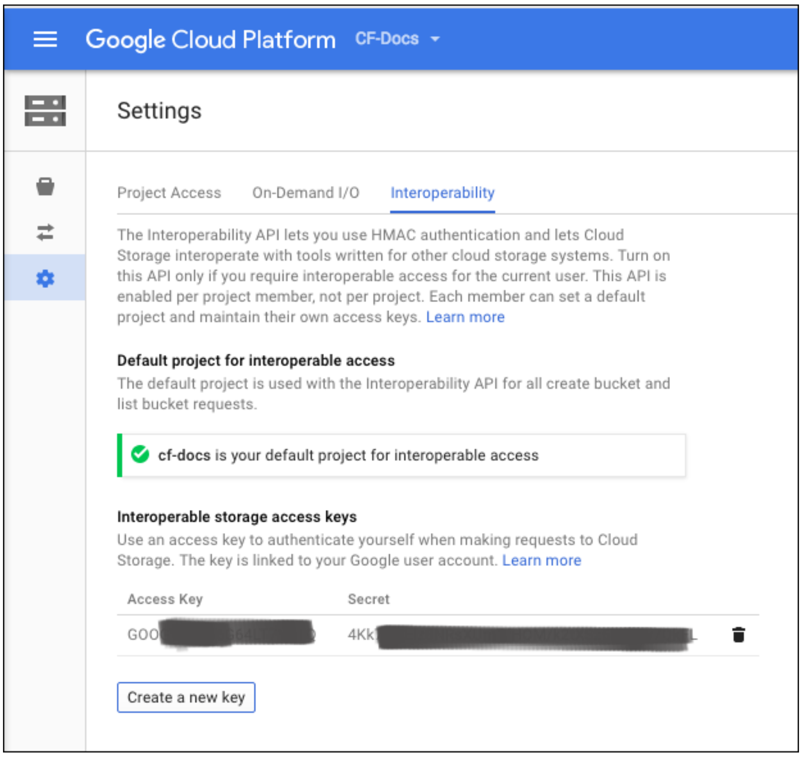
External Google Cloud Storage

To use external Google file storage for your Pivotal Application Service (PAS) filestore, perform the following steps:

1. Select the **External Google Cloud Storage** option.



2. Enter values for **Access Key** and **Secret Key**. To obtain the values for these fields:
   - In the GCP Console, navigate to the **Storage** tab, then click **Settings**.
   - Click **Interoperability**.
   - If necessary, click **Enable interoperability access**. If interoperability access is already enabled, confirm that the default project matches the project where you are installing PCF.



- Click **Create a new key**.
- Copy and paste the generated values into the corresponding PAS fields. PCF uses these values for authentication when connecting to Google Cloud Storage.

3. Enter the names of the storage buckets you created in [Preparing to Deploy PCF on GCP](#):
   - **Buildpacks Bucket Name**: Enter the value of `buildpacks_bucket` from your Terraform output.
   - **Droplets Bucket Name**: Enter the value of `droplets_bucket` from your Terraform output.
   - **Resources Bucket Name**: Enter the value of `packages_bucket` from your Terraform output.
   - **Packages Bucket Name**: Enter the value of `resources_bucket` from your Terraform output.
4. Click **Save**.

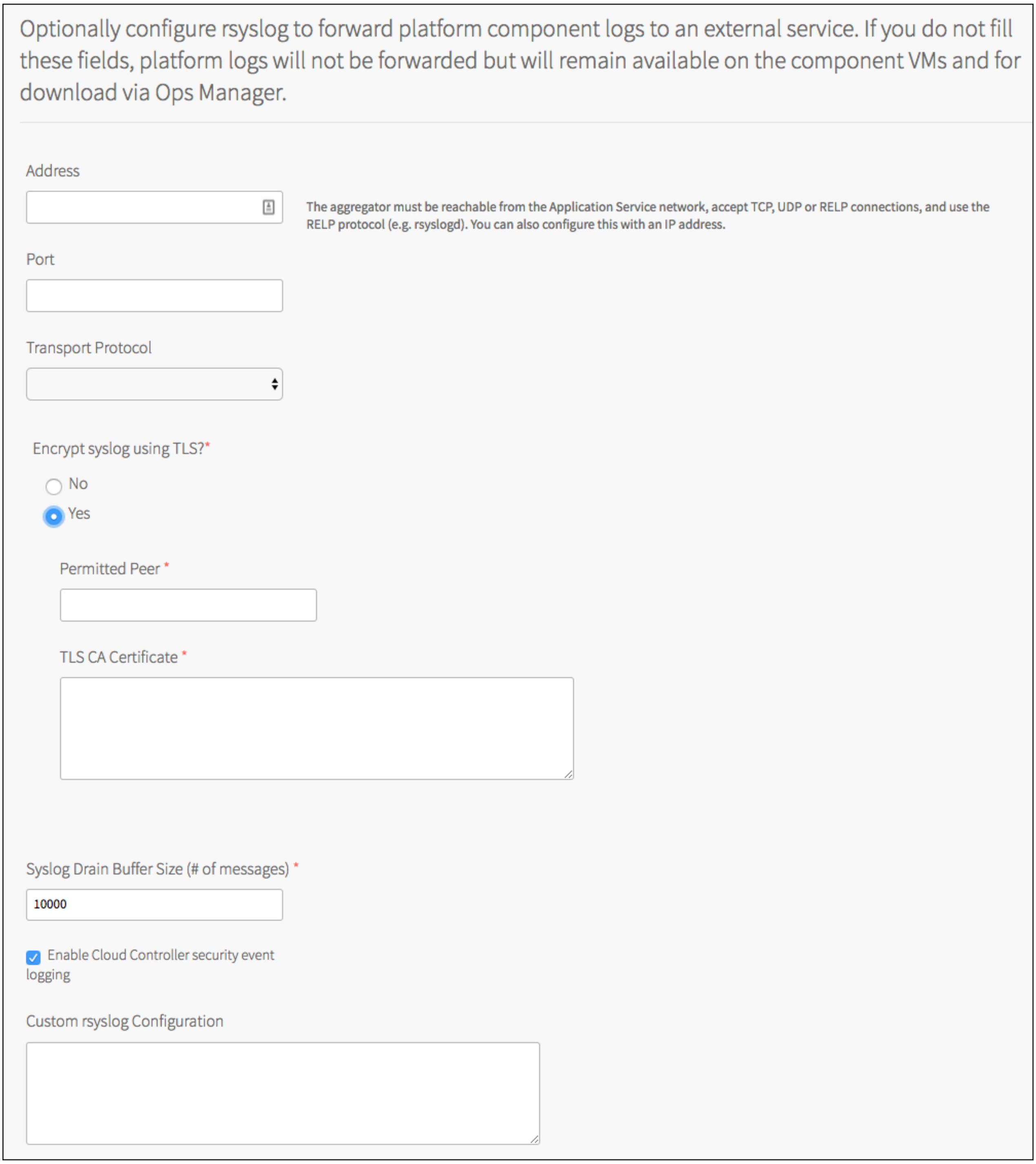
Other IaaS Storage Options

Azure Storage and External S3-Compatible File Storage are also available as file storage options, but Pivotal does not recommend these for a typical PCF on GCP installation.

Step 16: (Optional) Configure System Logging

If you forward logging messages to an external Reliable Event Logging Protocol (RELP) server, complete the following steps:

1. Select the **System Logging** section that is located within your PAS **Settings** tab.



2. Enter the IP address of your syslog server in **Address**.

3. Enter the port of your syslog server in **Port**. The default port for a syslog server is `514`.

> **Note**: The host must be reachable from the PAS network, accept TCP connections, and use the RELP protocol. Ensure your syslog server listens on external interfaces.

4. Select a **Transport Protocol** to use when forwarding logs.

5. If you plan to use TLS encryption when sending logs to the remote server, select **Yes** when answering the **Encrypt syslog using TLS?** question.

   a. In the **Permitted Peer** field, enter either the name or SHA1 fingerprint of the remote peer.
   b. In the **TLS CA Certificate** field, enter the TLS CA Certificate for the remote server.

6. For the **Syslog Drain Buffer Size**, enter the number of messages the Doppler server can hold from Metron agents before the server starts to drop them. See the Loggregator Guide for Cloud Foundry Operators topic for more details.

7. If you want to include security events in your log stream, select the **Enable Cloud Controller security event logging** checkbox. This logs all API requests, including the endpoint, user, source IP address, and request result, in the Common Event Format (CEF).

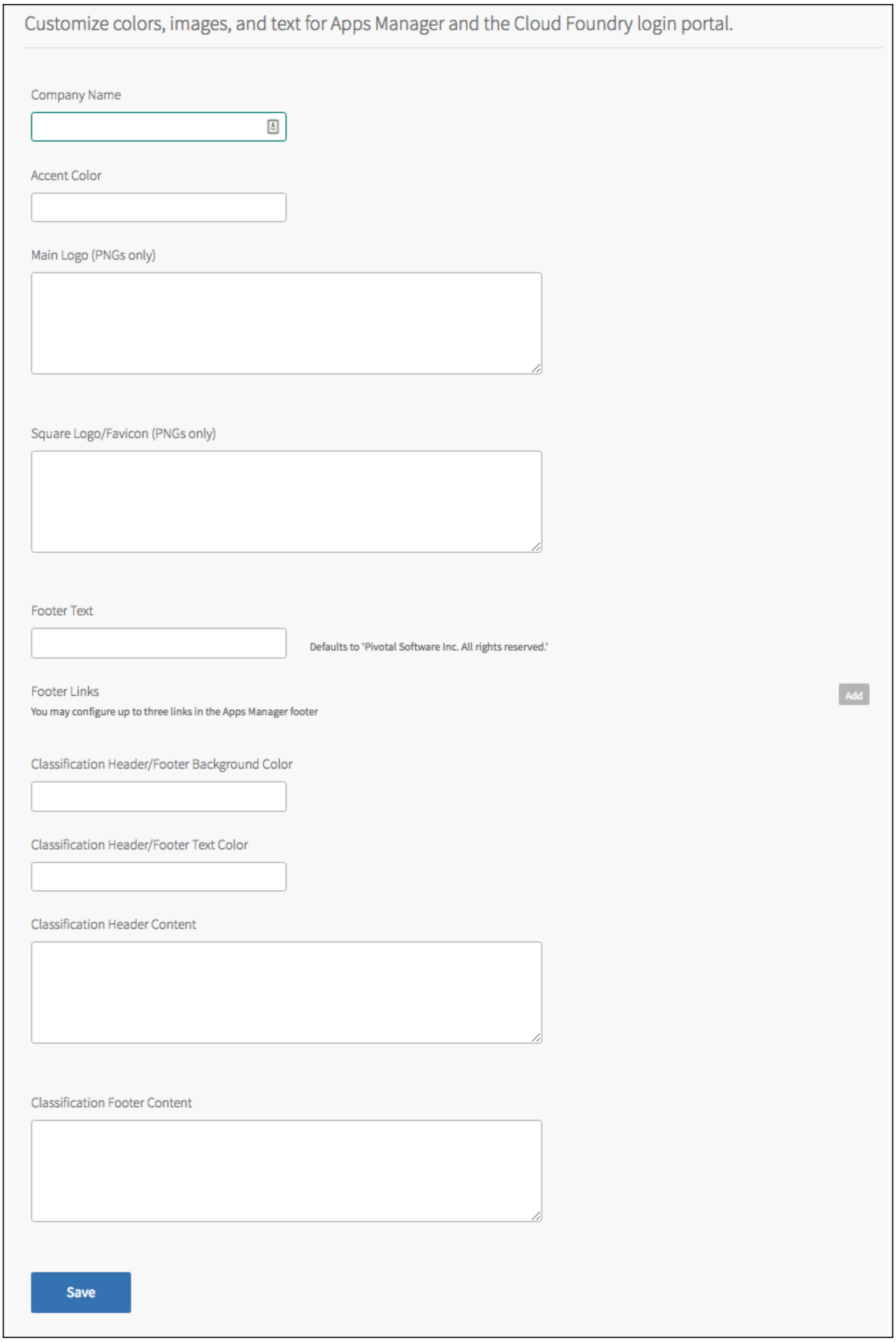
8. If you want to specify a custom syslog formatting rule, enter it in the **Custom rsyslog Configuration** field in RainerScript syntax.

9. Click **Save**.

Step 17: (Optional) Customize Apps Manager

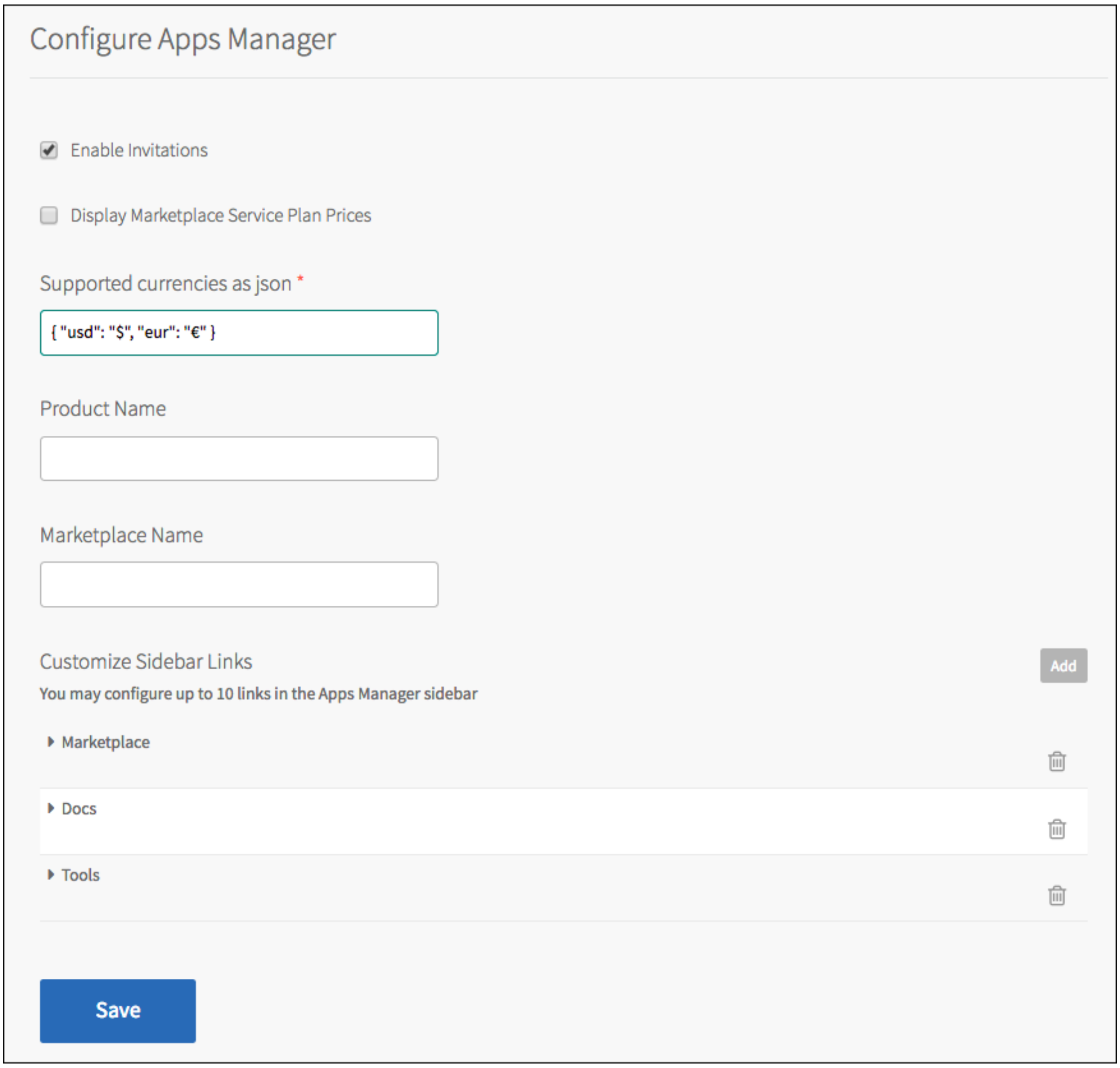
The **Custom Branding** and **Apps Manager** sections customize the appearance and functionality of Apps Manager. Refer to Custom Branding Apps Manager for descriptions of the fields on these pages and for more information about customizing Apps Manager.

1. Select **Custom Branding**. Use this section to configure the text, colors, and images of the interface that developers see when they log in, create an account, reset their password, or use Apps Manager.



2. Click **Save** to save your settings in this section.

3. Select **Apps Manager**.



4. Select **Enable Invitations** to enable invitations in Apps Manager. Space Managers can invite new users for a given space, Org Managers can invite new users for a given org, and Admins can invite new users across all orgs and spaces. See the Inviting New Users section of the *Managing User Roles with Apps Manager* topic for more information.

5. Select **Display Marketplace Service Plan Prices** to display the prices for your services plans in the Marketplace.

6. Enter the **Supported currencies as json** to appear in the Marketplace. Use the format `{"CURRENCY-CODE":"SYMBOL"}`. This defaults to `{"usd": "$", "eur": "€"}`.

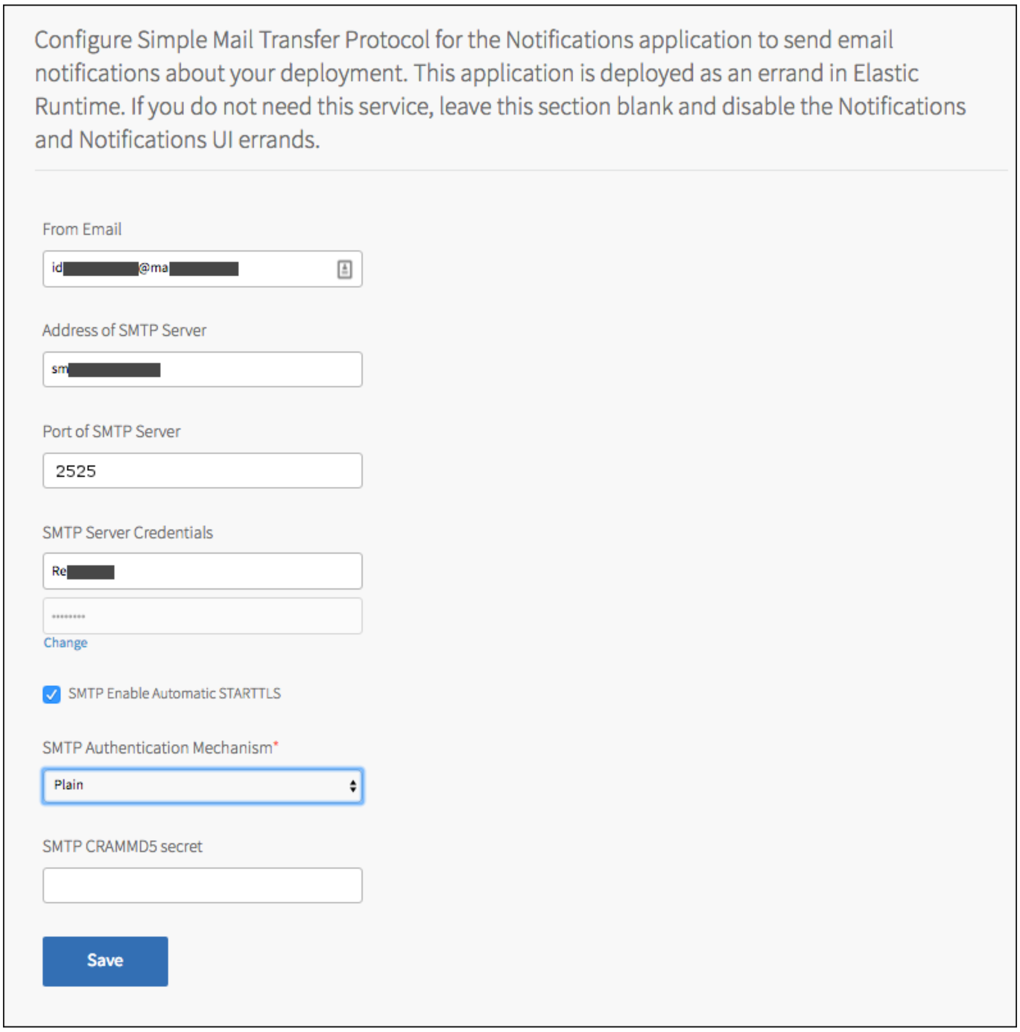
7. Use **Product Name**, **Marketplace Name**, and **Customize Sidebar Links** to configure page names and sidebar links in the **Apps Manager** and **Marketplace** pages.

8. Click **Save** to save your settings in this section.

Step 18: (Optional) Configure Email Notifications

PAS uses SMTP to send invitations and confirmations to Apps Manager users. You must complete the **Email Notifications** page if you want to enable end-user self-registration.

1. Select **Email Notifications**.



2. Enter your reply-to and SMTP email information. You must use port `2525`. Ports `25` and `587` are not allowed on GCP Compute Engine.

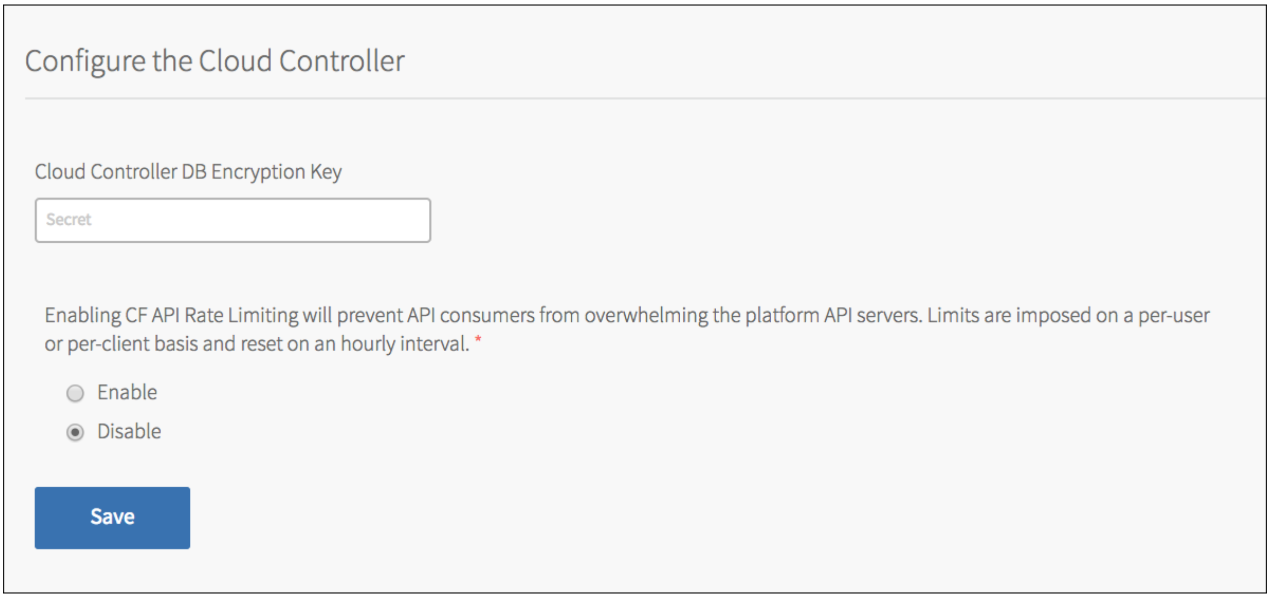
3. For **SMTP Authentication Mechanism**, select `none`.

4. Click **Save**.

> **Note**: If you do not configure the SMTP settings using this form, the administrator must create orgs and users using the cf CLI tool. See Creating and Managing Users with the cf CLI for more information.

Step 19: Configure Cloud Controller

1. Click **Cloud Controller**.



2. Enter your **Cloud Controller DB Encryption Key** if all of the following are true:
   - You deployed Pivotal Application Service (PAS) previously.
   - You then stopped PAS or it crashed.
   - You are re-deploying PAS with a backup of your Cloud Controller database.

   See Backing Up Pivotal Cloud Foundry for more information.

3. CF API Rate Limiting prevents API consumers from overwhelming the platform API servers. Limits are imposed on a per-user or per-client basis and reset on an hourly interval.

   To disable CF API Rate Limiting, select **Disable** under **Enable CF API Rate Limiting**. To enable CF API Rate Limiting, perform the following steps:

   a. Under **Enable CF API Rate Limiting**, select **Enable**.
   b. For **General Limit**, enter the number of requests a user or client is allowed to make over an hour interval for all endpoints that do not have a custom limit. The default value is `2000`.
   c. For **Unauthenticated Limit**, enter the number of requests an unauthenticated client is allowed to make over an hour interval. The default value is `100`.

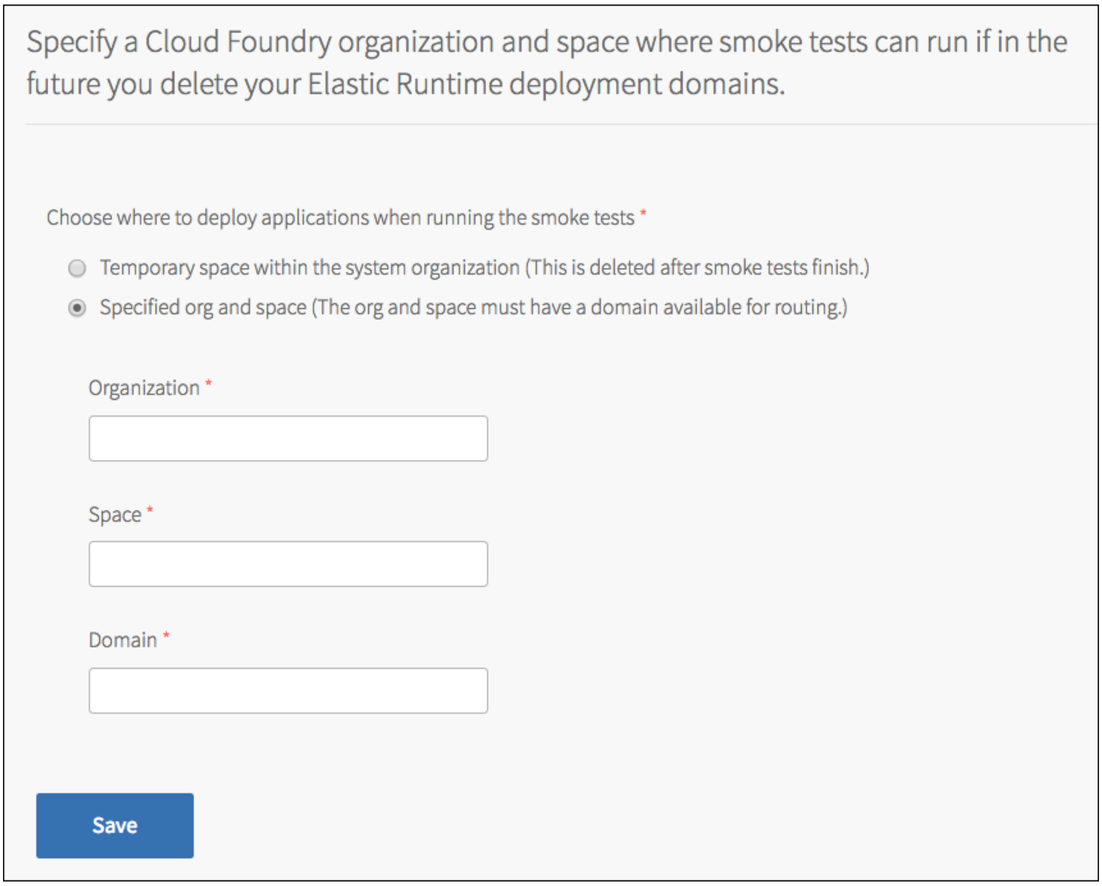
4. Click **Save**.

Step 20: Configure Smoke Tests

The Smoke Tests errand runs basic functionality tests against your Pivotal Application Service (PAS) deployment after an installation or update. In this section, choose where to run smoke tests. In the **Errands** section, you can choose whether or not to run the Smoke Tests errand.

1. Select **Smoke Tests**.

2. If you have a shared apps domain, select **Temporary space within the system organization**, which creates a temporary space within the `system` organization for running smoke tests and deletes the space afterwards. Otherwise, select **Specified org and space** and complete the fields to specify where you want to run smoke tests.



3. Click **Save**.

Step 21: (Optional) Enable Advanced Features

The **Advanced Features** section of Pivotal Application Service (PAS) includes new functionality that may have certain constraints. Although these features are fully supported, Pivotal recommends caution when using them in production environments.

Diego Cell Memory and Disk Overcommit

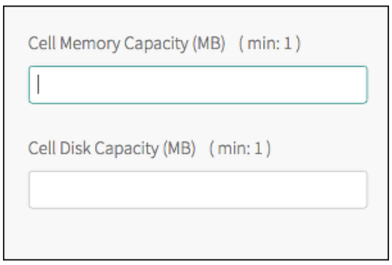
If your apps do not use the full allocation of disk space and memory set in the **Resource Config** tab, you might want use this feature. These fields control the amount to overcommit disk and memory resources to each Diego Cell VM.

For example, you might want to use the overcommit if your apps use a small amount of disk and memory capacity compared to the amounts set in the **Resource Config** settings for **Diego Cell**.

> **Note**: Due to the risk of app failure and the deployment-specific nature of disk and memory use, Pivotal has no recommendation about how much, if any, memory or disk space to overcommit.

To enable overcommit, follow these steps:

1. Select **Advanced Features**.



2. Enter the total desired amount of Diego cell memory value in the **Cell Memory Capacity (MB)** field. Refer to the **Diego Cell** row in the **Resource Config** tab for the current Cell memory capacity settings that this field overrides.

3. Enter the total desired amount of Diego cell disk capacity value in the **Cell Disk Capacity (MB)** field. Refer to the **Diego Cell** row in the **Resource Config** tab for the current Cell disk capacity settings that this field overrides.

4. Click **Save**.

> **Note**: Entries made to each of these two fields set the total amount of resources allocated, not the overage.

Whitelist for Non-RFC-1918 Private Networks

Some private networks require extra configuration so that internal file storage (WebDAV) can communicate with other PCF processes.

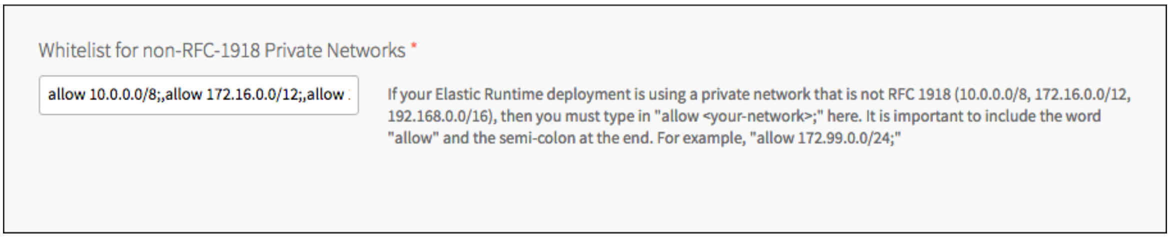
The **Whitelist for non-RFC-1918 Private Networks** field is provided for deployments that use a non-RFC 1918 private network. This is typically a private network other than `10.0.0.0/8`, `172.16.0.0/12`, or `192.168.0.0/16`.

Most PCF deployments do not require any modifications to this field.

To add your private network to the whitelist, perform the following steps:

1. Select **Advanced Features**.

2. Append a new `allow` rule to the existing contents of the **Whitelist for non-RFC-1918 Private Networks** field.

   Include the word `allow`, the network CIDR range to allow, and a semi-colon (`;`) at the end. For example: `allow 172.99.0.0/24;`

3. Click **Save**.

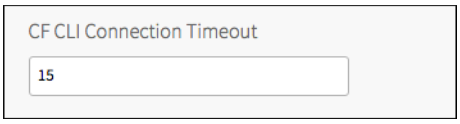
CF CLI Connection Timeout

The **CF CLI Connection Timeout** field allows you to override the default five second timeout of the Cloud Foundry Command Line Interface (cf CLI) used within your PCF deployment. This timeout affects the cf CLI command used to push PAS errand apps such as Notifications, Autoscaler, and Apps Manager.

Set the value of this field to a higher value, in seconds, if you are experiencing domain name resolution timeouts when pushing errands in PAS.

To modify the value of the **CF CLI Connection Timeout**, perform the following steps:

1. Select **Advanced Features**.

2. Add a value, in seconds, to the **CF CLI Connection Timeout** field.
   

3. Click **Save**.

Step 22: Configure Errands

Errands are scripts that Ops Manager runs automatically when it installs or uninstalls a product, such as a new version of Pivotal Application Service (PAS). There are two types of errands: *post-deploy errands* run after the product is installed, and *pre-delete errands* run before the product in uninstalled.

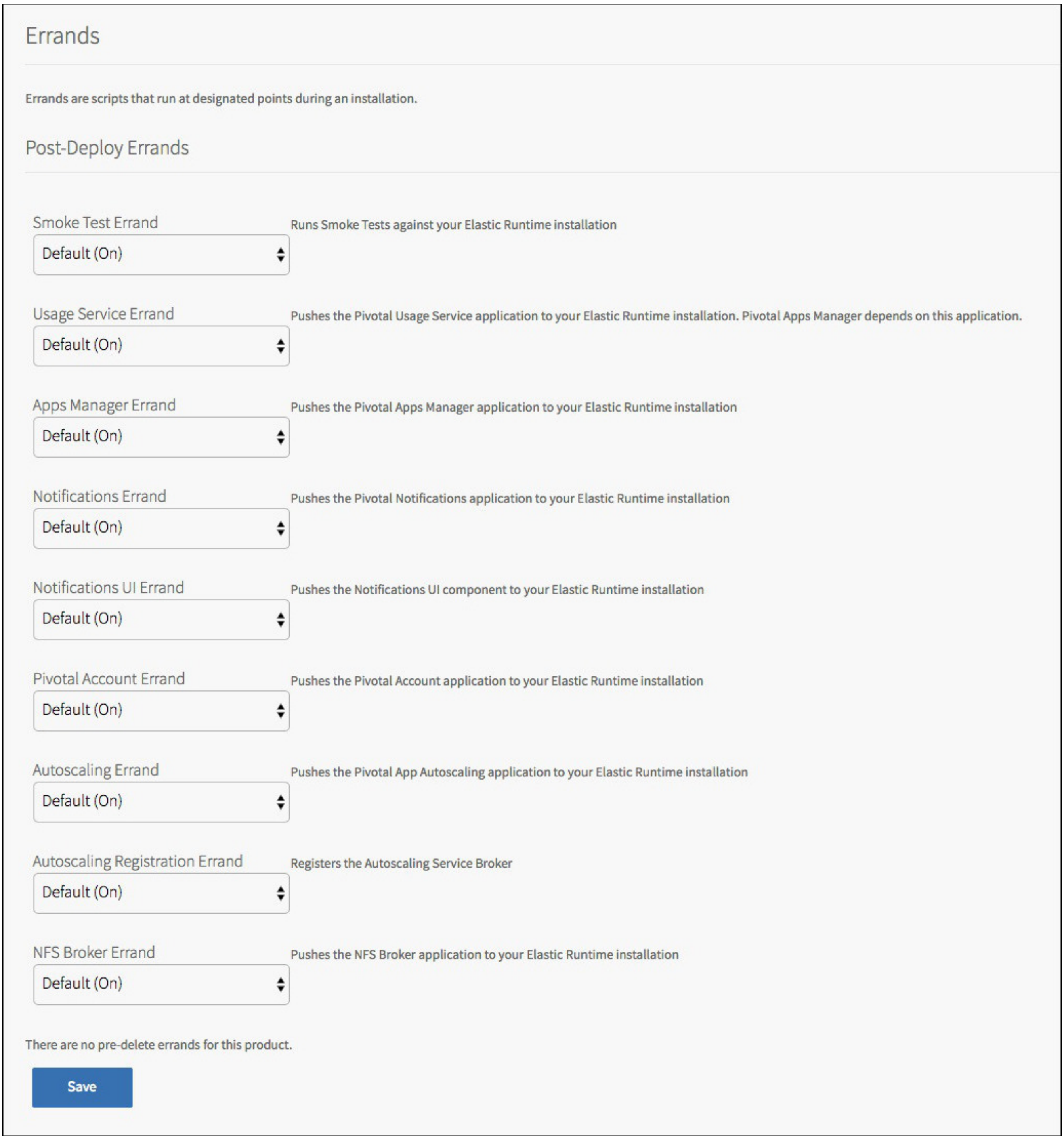
By default, Ops Manager always runs pre-delete errands, and only runs post-deploy errands when the product has changed since the last time Ops

Manager installed something. In PAS, the Smoke Test Errand defaults to always run.

The PAS tile **Errands** pane lets you change these run rules. For each errand, you can select **On** to run it always, **Off** to never run it, or **When Changed** to run it only when the product has changed since the last install.

For more information about how Ops Manager manages errands, see the [Managing Errands in Ops Manager] topic.

> **Note**: Several errands deploy apps that provide services for your deployment, such as Autoscaling and Notifications. Once one of these apps is running, selecting **Off** for the corresponding errand on a subsequent installation does not stop the app.



- **Smoke Test Errand** verifies that your deployment can do the following:
  - Push, scale, and delete apps
  - Create and delete orgs and spaces
- **Usage Service Errand** deploys the Pivotal Usage Service application, which Apps Manager depends on.
- **Apps Manager Errand** deploys Apps Manager, a dashboard for managing apps, services, orgs, users, and spaces. Until you deploy Apps Manager, you

must perform these functions through the cf CLI. After Apps Manager has been deployed, Pivotal recommends deselecting the checkbox for this errand on subsequent PAS deployments. For more information about Apps Manager, see the Getting Started with the Apps Manager topic.

- **Notifications Errand** deploys an API for sending email notifications to your PCF platform users.

> **Note**: The Notifications app requires that you configure SMTP with a username and password, even if you set the value of **SMTP Authentication Mechanism** to `none`.

- **Notifications UI Errand** deploys a dashboard for users to manage notification subscriptions.
- **Pivotal Account Errand** deploys Pivotal Account, a dashboard that allows users to create and manage their accounts. In the Pivotal Account dashboard, users can launch applications, manage their profiles, manage account security, manage notifications, and manage approvals. See the Enabling Pivotal Account topic for more information.
- **Autoscaling Errand** enables you to configure your apps to automatically scale in response to changes in their usage load. See the Scaling an Application Using Autoscaler topic for more information.
- **Autoscaling Registration Errand** makes the Autoscaling service available to your applications. Without this errand, you cannot bind the Autoscaling app to your apps.
- **NFS Broker Errand** enables you to use NFS Volume Services by installing the NFS Broker app in PAS. See the Enabling NFS Volume Services topic for more information.

Step 23: Configure Load Balancers

1. Under the **LOAD BALANCERS** column of the **Router** row, enter a comma-delimited list consisting of the values of `ws_router_pool` and `http_lb_backend_name` from your Terraform output. For example, `tcp:pcf-cf-ws,http:pcf-httpslb`. These are the names of the TCP WebSockets and HTTP(S) load balancers for your deployment.

   > **Note**: Do not add a space between key/value pairs in the `LOAD BALANCER` field or it will fail.

   > **Note**: If you are using HAProxy in your deployment, then enter the above load balancer values in the `LOAD BALANCERS` field of the **HAPRoxy** row instead of the **Router** row. For a high availability configuration, scale up the HAProxy job to more than one instance.

2. If you have enabled TCP routing in the Networking pane, add the value of `tcp_router_pool` from your Terraform output, prepended with `tcp:`, to the **LOAD BALANCERS** column of the TCP Router row. For example, `tcp:pcf-cf-tcp`.

3. Enter the name of you SSH load balancer depending on which release you are using.
   - **PAS**: Under the **LOAD BALANCERS** column of the **Diego Brain** row, enter the value of `ssh_router_pool` from your Terraform output, prepended with `tcp:`. For example, `tcp:MY-PCF-ssh-proxy`.
   - **Small Footprint Runtime**: Under the **LOAD BALANCERS** column of the **Control** row, enter the value of `ssh_router_pool` from your Terraform output, prepended with `tcp:`.

4. Verify that the **Internet Connected** checkbox for every job is checked. The terraform templates do not provision a Network Address Translation (NAT) box for Internet connectivity to your VMs so instead they will be provided with ephemeral public IP addresses to allow the jobs to reach the Internet.

   > **Note**: If you want to provision a Network Address Translation (NAT) box to provide Internet connectivity to your VMs instead of providing them with public IP addresses, deselect the **Internet Connected** checkboxes. For more information about using NAT in GCP, see the GCP documentation.

5. Click **Save**.

Step 24: (Optional) Scale Down and Disable Resources

> **Note**: The **Resource Config** pane has fewer VMs if you are installing the Small Footprint Runtime.

> **Note**: The Small Footprint Runtime does not default to a highly available configuration. It defaults to the minimum configuration. If you want to make the Small Footprint Runtime highly available, scale the **Compute**, **Router**, and **Database** VMs to `3` instances and scale the **Control** VM to `2` instances.

Pivotal Application Service (PAS) defaults to a highly available resource configuration. However, you may need to perform additional procedures to make your deployment highly available. See the Zero Downtime Deployment and Scaling in CF and the Scaling Instances in PAS topics for more information.

If you do not want a highly available resource configuration, you must scale down your instances manually by navigating to the **Resource Config** section and using the drop-down menus under **Instances** for each job.

By default, PAS also uses an internal filestore and internal databases. If you configure PAS to use external resources, you can disable the corresponding system-provided resources in Ops Manager to reduce costs and administrative overhead.

Complete the following procedures to disable specific VMs in Ops Manager:

1. Click **Resource Config**.

2. If you configure PAS to use an external S3-compatible filestore, edit the following fields:
    - **File Storage**: Enter `0` in **Instances**.

3. If you selected **External** when configuring the UAA or System databases, edit the following fields:
    - **MySQL Proxy**: Enter `0` in **Instances**.
    - **MySQL Server**: Enter `0` in **Instances**.
    - **MySQL Monitor**: Enter `0` in **Instances**.
    - **Cloud Controller Database**: Enter `0` in **Instances**.
    - **UAA Database**: Enter `0` in **Instances**.

4. If you are not using HAProxy, enter `0` in the **Instances** field for **HAProxy**.

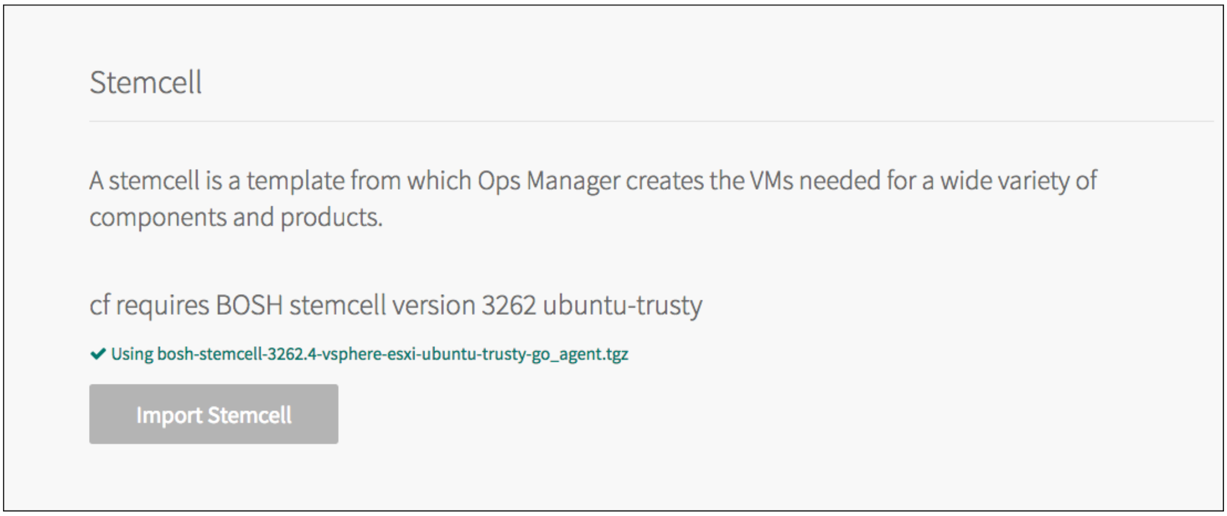
5. Click **Save**.

Step 25: Verify and Download Stemcell Version

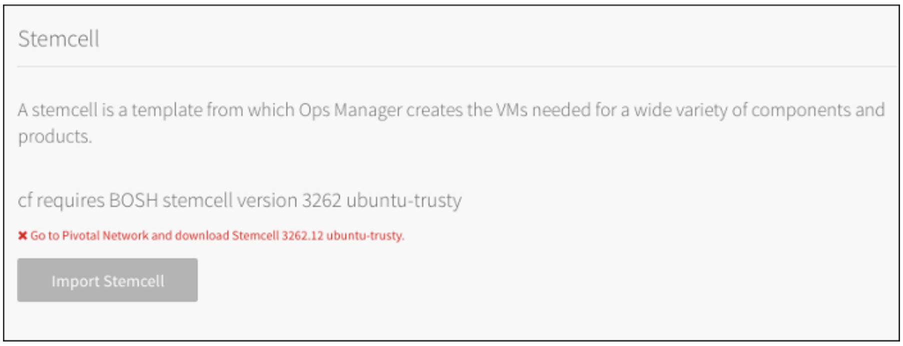
Verify whether Ops Manager is providing the stemcell version required by PAS. If the correct version is already present, you do not need to download a new stemcell.

1. In the PAS tile, select **Stemcell**.

2. Verify that the version indicated in the filename matches the version of stemcell required by PAS.
    - If PAS detects that a stemcell `.tgz` file is present in the Ops Manager Director VM at `/var/tempest/stemcells/`, the Stemcell screen displays filename information.



    - If PAS cannot detect a stemcell `.tgz` file, the following message displays:



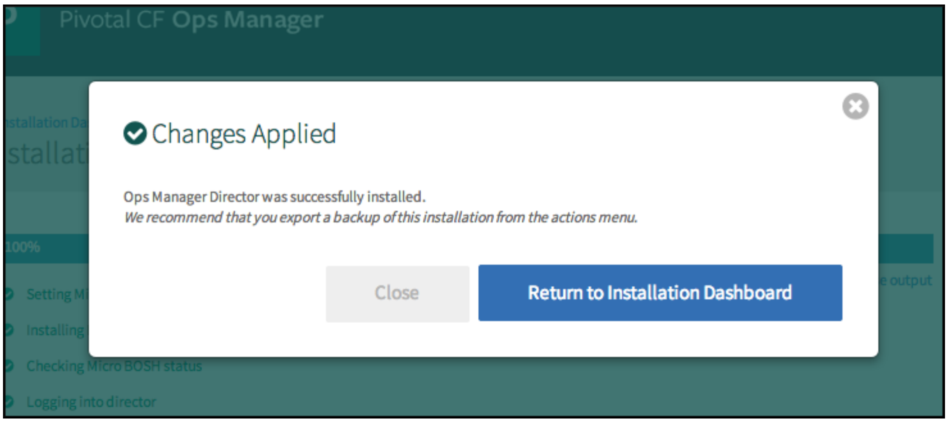
3. If the version of the stemcell file that is loaded does not match the required version listed in the Pivotal Network download page for PAS, or cannot be found by Ops Manager, perform the following steps to download and import a new stemcell file:

   a. Log in to the Pivotal Network and click **Stemcells**.
   b. Download the appropriate stemcell version targeted for your IaaS.
   c. In the **Stemcell** section of the PAS tile, click **Import Stemcell** to import the downloaded stemcell `.tgz` file.

Step 26: Complete the PAS Installation

1. Click the **Installation Dashboard** link to return to the Installation Dashboard.

2. Click **Apply Changes**.
   The install process generally requires a minimum of 90 minutes to complete. The image shows the Changes Applied window that displays when the installation process successfully completes.



Configuring a Shared VPC on GCP

Page last updated:

This guide describes the preparation steps required to configure and integrate a shared Virtual Private Cloud (VPC) on Google Cloud Platform (GCP) with Pivotal Cloud Foundry (PCF).

GCP Shared VPC, formerly known as Google Cross-Project Networking (XPN), enables you to assign GCP resources to individual projects within an organization but allows communication and shared services between projects. For more information about shared VPCs, see Shared VPC Overview in the GCP documentation.

Prerequisites

To configure a shared VPC, you must assign your project to a Cloud Organization. Confirm that you have a Cloud Organization associated with your GCP account using one of the following methods:

- **GCP Console**: From https://console.cloud.google.com, click the **Organization** drop-down menu at the top of the page to display all organizations you belong to.
- **gcloud Command Line Interface (CLI)**: From the command line, run `gcloud organizations list` to display all organizations you belong to. See gcloud Overview in the Google documentation to install the gcloud CLI.

For more information, see Creating and Managing Organizations in the GCP documentation. If you do not have a Cloud Organization, contact GCP support.

Step 1: Provision the Shared VPC

Follow the Enabling a shared VPC host project procedure in the GCP documentation. This procedure requires shared VPC admin permissions.

Step 2: Create a Shared VPC Network

Use the procedures in the Preparing to Deploy PCF on GCP topic to create a new network with firewall rules. Complete the following steps:

- Step 3: Create a GCP Network with Subnet
- Step 5: Create Firewall Rules for the Network

Step 3: Connect the Shared VPC to Ops Manager

You can use the GCP console or the gcloud CLI to connect the shared VPC host project with Ops Manager.

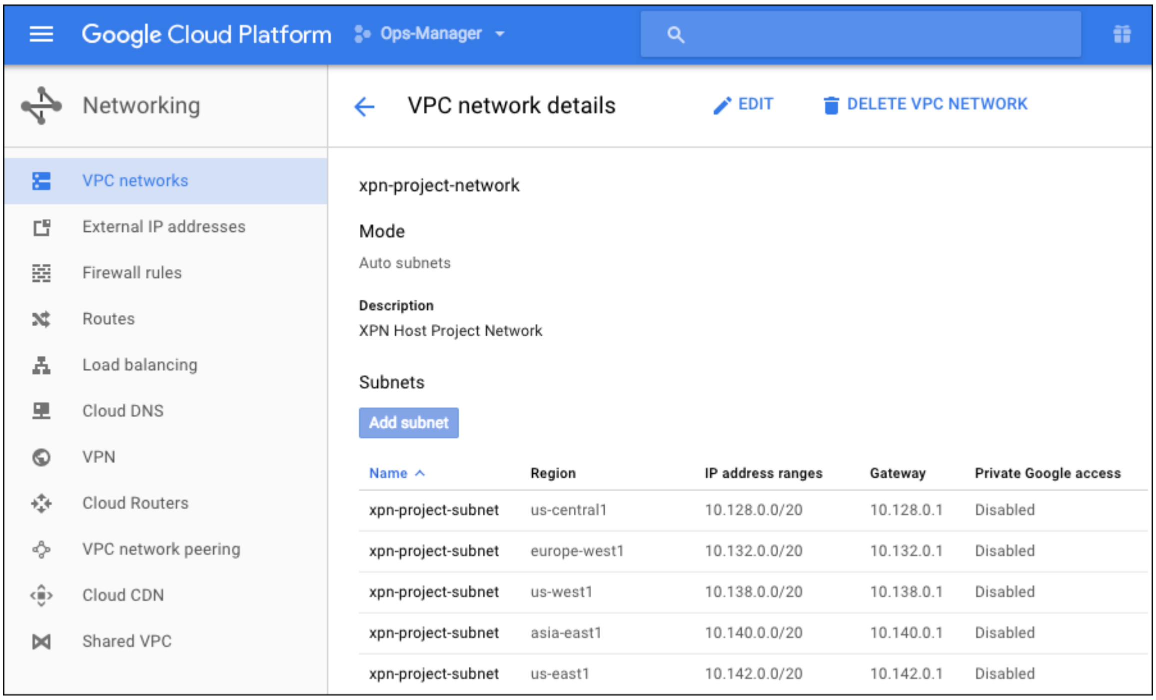
For more information, see VPC Network Peering in the GCP documentation.

WARNING: VPC Network Peering is currently in beta and intended for evaluation and test purposes only.

Set Up VPC Network Peering with GCP Console

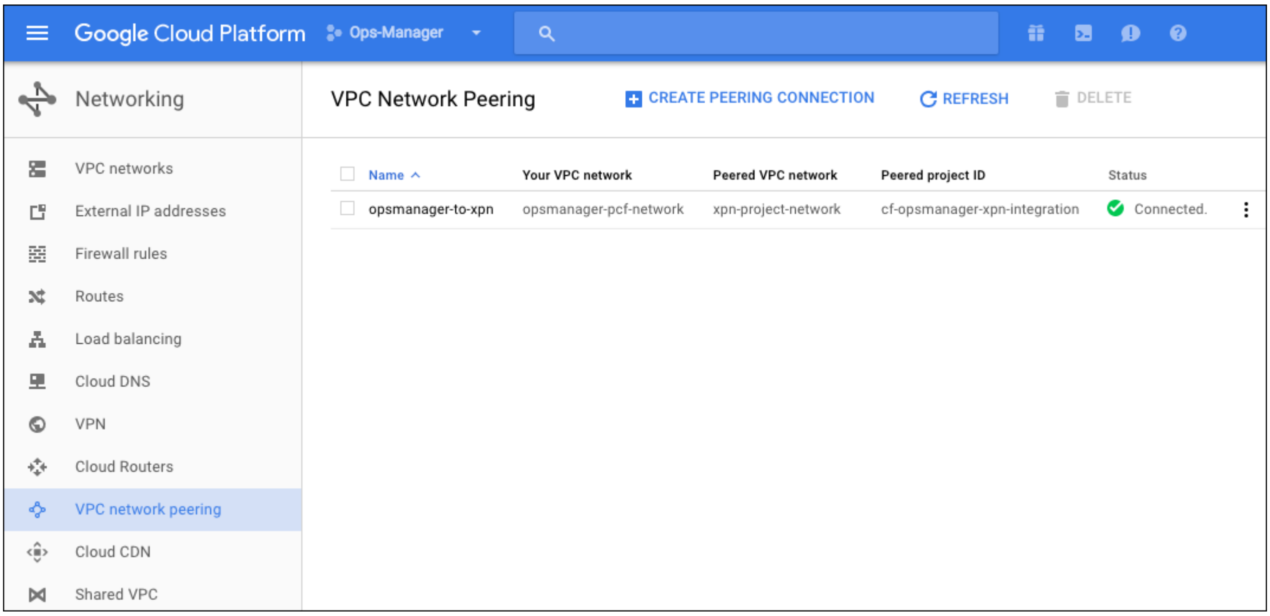
To set up VPC network peering with the GCP console, perform the following steps:

1. From https://console.cloud.google.com, click **Networking**, then **VPC networks**.



2. Click **Create Peering Connection**.

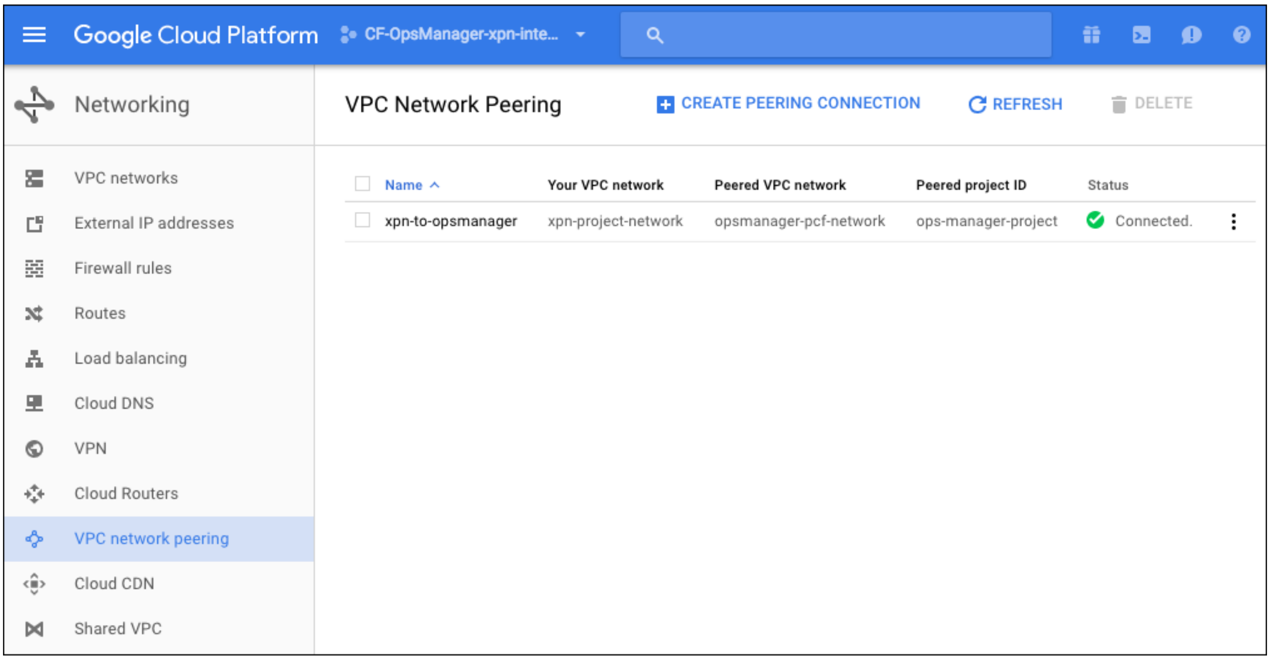
3. Enter a name for the network connection from the Ops Manager project to the new shared network, such as `opsmanager-to-xpn`.



4. Click **Save**.

5. Click **Create Peering Connection**.

6. Enter a name for the network connection from the new shared network to the Ops Manager project, such as `xpn-to-opsmanager`.



7. Click **Save**.

Set Up VPC Network Peering with gcloud CLI

To set up VPC network peering with the gcloud CLI, perform the following steps:

1. Enter the following command, replacing `OPSMANAGER-PROJECT` with the name of the project that contains your Ops Manager installation:

```
$ gcloud config set project OPSMANAGER-PROJECT
```

2. Enter the following command to create a connection from the Ops Manager project to the new shared VPC project:

```
$ gcloud beta compute networks peerings create OPSMANAGER-TO-VPC \
    --network OPSMANAGER-NETWORK \
    --peer-project VPC-HOST-PROJECT \
    --peer-network VPC-NETWORK \
    --auto-create-routes
```

   Replace the following text in the command above:
   - `OPSMANAGER-TO-VPC` : Choose a name for the connection, such as `om-to-vpc` .
   - `OPSMANAGER-NETWORK` : Enter the name of the network assigned to the Ops Manager project in GCP, such as `my-om-project` .
   - `VPC-HOST-PROJECT` : Enter the name you gave the shared VPC project in Step 1: Provision the Shared VPC.
   - `VPC-NETWORK` : Enter the name of the network you gave the shared VPC project in Step 2: Create Shared VPC Networks.

3. Enter the following command, replacing `VPC-HOST-PROJECT` with the new shared VPC project you created in Step 1: Provision the Shared VPC:

```
$ gcloud config set project VPC-HOST-PROJECT
```

4. Enter the following command to create a connection from the new shared VPC project to the Ops Manager project:

```
$ gcloud beta compute networks peerings create VPC-TO-OPSMANAGER \
    --network VPC-NETWORK \
    --peer-project OPSMANAGER-PROJECT \
    --peer-network OPSMANAGER-NETWORK \
    --auto-create-routes
```

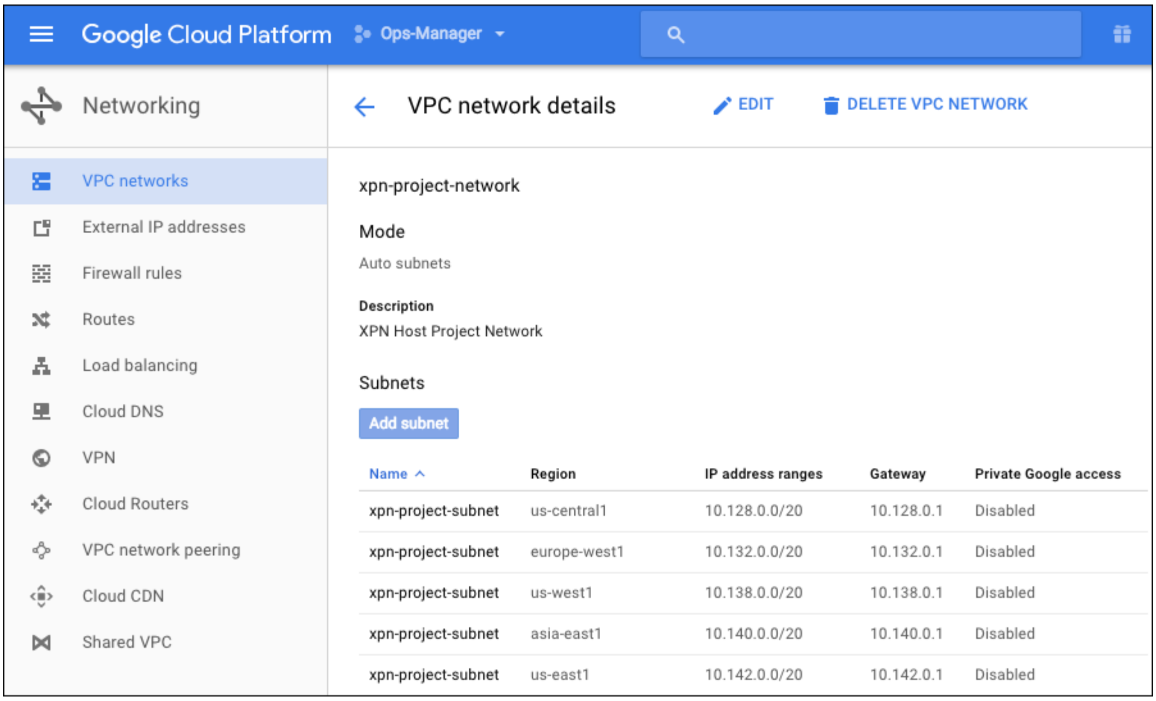
   Replace the following text and run the following command:
   - `VPC-TO-OPSMANAGER` : Choose a name for the connection, such as `vpc-to-om` .
   - `VPC-NETWORK` : Enter the name of the network you gave the shared VPC project in Step 2: Create Shared VPC Networks.
   - `OPSMANAGER-PROJECT` : Enter the name of the project that contains your Ops Manager installation.
   - `OPSMANAGER-NETWORK` : Enter the name of the network assigned to the Ops Manager project in GCP.

Step 4: Verify the Shared VPC Configuration

After configuring a shared VPC, use the following procedure to verify that the shared VPC host project VM appears in the Ops Manager project.

1. From https://console.cloud.google.com, select the Ops Manager project from the drop-down menu at the top of the page.

2. Click **Networking**, then **VPC networks**.



3. Confirm that the shared VPC network name appears in the **Subnets** list.

4. Confirm that the shared VPC network **IP address ranges** match what you set for the new VPC project in Step 2: Create a Shared VPC Network.

Deleting PCF from GCP

Page last updated:

When you deploy Pivotal Cloud Foundry (PCF) to Google Cloud Platform (GCP), you provision a set of resources. This topic describes how to delete the resources associated with a PCF deployment.

You can delete the resources in one of two ways:

- If you created a separate project for your PCF deployment, perform the procedure in the Delete the Project delete the project.
- If the project that contains your PCF deployment also contains other resources that you want to preserve, perform the procedure in the Delete PCF Resources section.

Delete the Project

Perform the following steps to delete the project for your PCF deployment:

1. Navigate to the GCP Console Dashboard.
2. Under your **Project**, click **Manage project settings**.
3. Click **DELETE PROJECT**.
4. Enter your project ID and click **SHUT DOWN** to confirm.

Delete PCF Resources

Perform the following steps to delete the resources associated with your PCF deployment:

1. Navigate to the GCP Console Dashboard.
2. Click the upper left icon and select **Networking**.
3. Click **Load balancing**.
4. Perform the following steps for all load balancers associated with your PCF deployment:
    a. Click the trashcan icon next to the load balancer.
    b. In the next dialog, select any **health checks** and **backend services** associated with the load balancer.
    c. Click **DELETE LOAD BALANCER AND THE SELECTED RESOURCES**.
5. Click the upper left icon and select **Compute Engine**.
6. Perform the following steps for **VM instances**, **Instance groups**, and **Disks**:
    a. Select the checkbox next to the PCF resource.
    b. When all PCF resources are selected, click **DELETE** in the upper right.
    c. Click **DELETE** to confirm.
7. Click the upper left icon and select **Networking**.
8. Click **External IP addresses**.
9. Select all external IP addresses associated with your PCF deployment, and click **RELEASE STATIC ADDRESS**.
10. Click on **Networks**, and perform the following steps for any networks you created for PCF:
    a. Click the name of the network.
    b. Click **DELETE NETWORK**.
    c. Click **DELETE** to confirm.
11. Click the upper left icon and select **IAM & Admin**.
12. Click the trashcan icon next to the `bosh` service account you created for PCF and click **REMOVE**.

13. Navigate to **Compute Engine > Metadata > SSH Keys**. Delete the `vcap` SSH key that you created for the project.

Troubleshooting PCF on GCP

Page last updated:

This topic describes how to troubleshoot known issues when deploying Pivotal Cloud Foundry (PCF) on Google Cloud Platform (GCP).

Problems Connecting with Single Sign-On (SSO)

Users may be unable to connect to applications running on PCF using SSO.

Explanation

SSO does not support multi-subnets.

Solution

Ensure that you have configured only one subnet. See the Preparing the GCP Environment for Deployment topic for information.

Uploading PAS Tile Causes Ops Manager Rails Application Crash

Uploading the Pivotal Application Service (PAS) tile causes the Ops Manager Rails application to crash.

Explanation

In compressed format, the PAS tile is 5GB. However, when uncompressed during installation, the PAS tile requires additional memory that can exhaust the memory allocated to the boot disk.

Solution

Ensure that the boot disk is allocated at least 50GB of memory. See Step 3: Create the Ops Manager VM Instance for more information.

Problems Deploying Diego for Windows

Deploying Diego for Windows as described in fails with a `PSSecurity Unauthorized Access` error.

For example:

```
.\setup.ps1 : File C:\Users\username\Downloads\DiegoWindows\setup.ps1
cannot be loaded. The file C:\Users\username\Downloads\DiegoWindows\setup.ps1
is not digitally signed. You cannot run this script on the current system.
For more information about running scripts and setting execution policy, see
about_Execution_Policies at http://go.microsoft.com/fwlink/?LinkID=135170.
At line:1 char:1
+ .\setup.ps1
+ ~~~~~~~~~~~
    + CategoryInfo          : SecurityError: (:) [], PSSecurityException
    + FullyQualifiedErrorId : UnauthorizedAccess
```

Explanation

On GCP, deploying Diego for Windows requires elevated PowerShell privileges.

Solution

As a workaround, execute the following cmdlet before running the `setup.ps1` script:

```
Set-ExecutionPolicy Unrestricted
```

For more information about this cmdlet, see Using the Set-ExecutionPolicy Cmdlet.

PAS Deployment Fails - MySQL Monitor replication-canary Job

During installation of the PAS tile, the replication-canary job fails to start. The error reported in the installation log resembles the following:

```
Started updating job mysql_monitor > mysql_monitor/0
(48e7ec82-3cdf-41af-9d0f-90d1f12683c8) (canary). Failed: 'mysql_monitor/0
(48e7ec82-3cdf-41af-9d0f-90d1f12683c8)' is not running after update.
Review logs for failed jobs: replication-canary (00:05:13)

Error 400007: 'mysql_monitor/0 (48e7ec82-3cdf-41af-9d0f-90d1f12683c8)'
is not running after update.
Review logs for failed jobs: replication-canary
```

Explanation

This error can appear as a result of incorrect configuration of network traffic and missed communication between the Gorouter and a load balancer.

Solution

1. Make sure you have selected the **Forward SSL to PAS Router** option in your PAS Network Configuration.
2. Verify that you have configured the firewall rules properly and that TCP ports `80`, `443`, `2222`, and `8080` are accessible on your GCP load balancers. See Create Firewall Rules for the Network.
3. Verify that you have configured the proper SSL certificates on your HTTP(S) load balancer in GCP.
4. If necessary, re-upload a new certificate and update any required SSL Certificate and SSH Key fields in your PAS network configuration.

Upgrading Ops Manager Director on GCP

Page last updated:

This topic describes how to upgrade Ops Manager Director for Pivotal Cloud Foundry (PCF) on Google Cloud Platform (GCP).

In this procedure, you create a new Ops Manager VM instance that hosts the new version of Ops Manager. To upgrade, you export your existing Ops Manager installation into this new VM.

After you complete this procedure, follow the instructions in Upgrading PAS and Other Pivotal Cloud Foundry Products.

Step 1: Locate the Pivotal Ops Manager Installation File

1. Log in to the Pivotal Network, and click on **Pivotal Cloud Foundry Operations Manager**.

2. From the **Releases** drop-down, select the release for your upgrade.

3. Select one of the following download files:
   - **Pivotal Cloud Foundry Ops Manager for GCP**
   - **Pivotal Cloud Foundry Ops Manager YAML for GCP** When you click on the download link, your browser downloads or opens the `OpsManager_version_onGCP.pdf` or `OpsManager_version_onGCP.yml` file.

   These documents provide the GCP location of the Ops Manager `.tar.gz` installation file based on the geographic location of your installation.

4. Copy the filepath string of the Ops Manager image based on your existing deployment location.

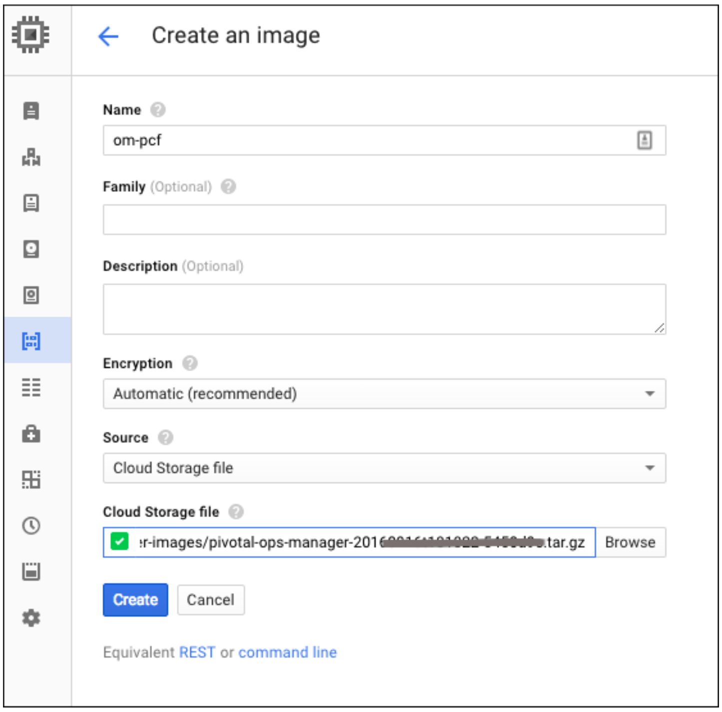
Step 2: Create a Private VM Image

1. Log in to the GCP Console.

2. In the left navigation panel, click **Compute Engine**, and select **Images**.

3. Click **Create Image**.

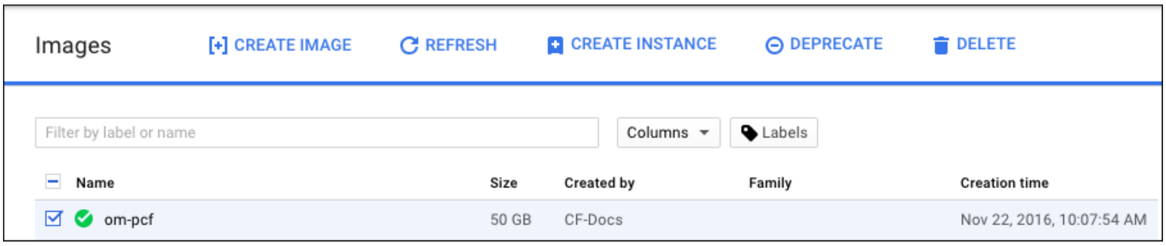
4. Complete the following fields:
   - **Name**: Enter a name that matches the naming convention of your existing Ops Manager image files.
   - **Encryption**: Leave **Automatic (recommended)** selected.
   - **Source**: Choose **Cloud Storage file**.
   - **Cloud Storage file**: Paste in the Google Cloud Storage filepath you copied from the PDF or YAML file in the previous step.



5. Click **Create**. The file may take a few minutes to import.

Step 3: Create the Ops Manager VM Instance

1. Select the checkbox for the image that you created above.



2. Click **Create Instance**.

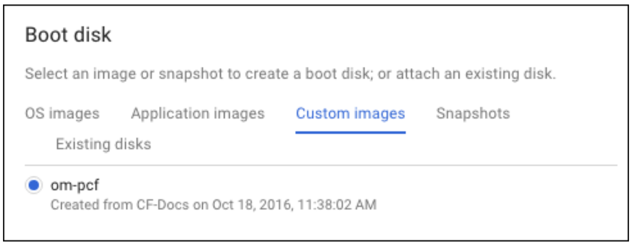
3. In the **Create an instance form**, complete the following fields:
   - **Name**: Enter a name that matches the naming conventions of your existing deployment.
   - **Zone**: Choose a zone from the region of your existing deployment.
   - **Machine type**: Click **Customize** to manually configure the vCPU and memory. An Ops Manager VM instance requires the following minimum specifications:

Machine Spec	Minimum Value
CPU	2 vCPUs
Memory	8 GB

   - **Boot disk**: Click **Change**, then perform the following steps:
     - Click **Custom images** if it is not already selected.
     - Select the **Boot disk type**. If you have an Ops Manager environment with high performance needs, select **SSD**. As an example, environments used to develop PCF tiles may benefit from a higher performing Ops Manager VM boot disk. For most environments, however, you can select **Standard**.
     - Set the **Size (GB)** of the boot disk to the minimum or higher.

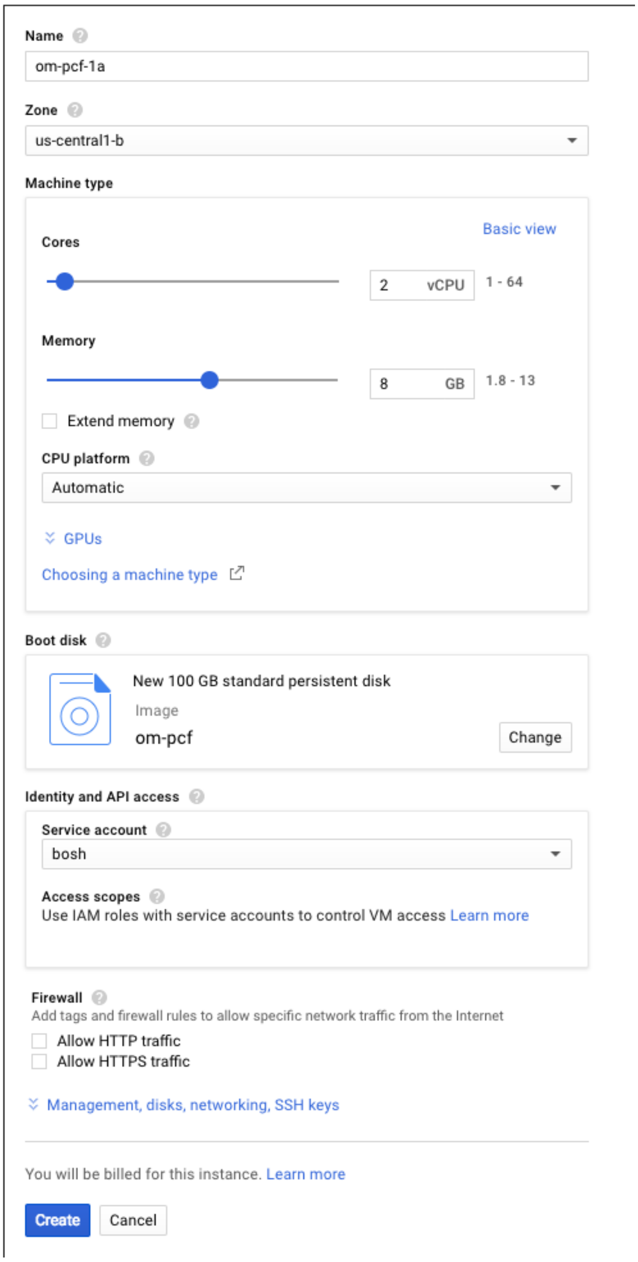
Machine Spec	Minimum Value
Boot disk	100 GB

     - Select the Ops Manager image you created in the previous step if it is not already selected.



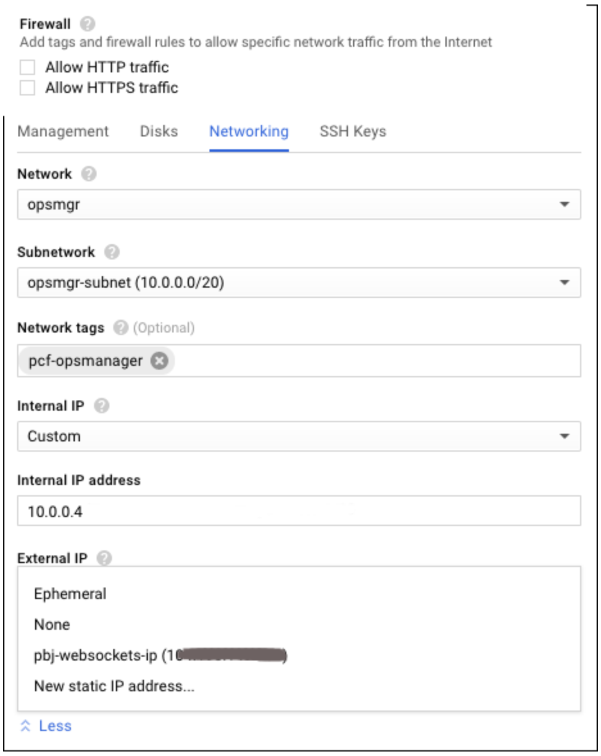
- Click **Select** to save.

- Under **Identity and API access**, choose the **Service account** you created when you initially installed Pivotal Cloud Foundry. See Set up an IAM Service Account.
- **Allow HTTP traffic**: Only select this checkbox if you selected it in your original Ops Manager VM configuration.
- **Allow HTTPS traffic**: Only select this checkbox if you selected it in your original Ops Manager VM configuration.



- **Networking**: Select the **Networking** tab, and perform the following steps:
  - For **Network** and **Subnetwork**, select the network and subnetwork you created when you initially deployed Pivotal Cloud Foundry. See Create a GCP Network with Subnet section of the *Preparing to Deploy PCF on GCP* topic.
  - For **Network tags**, enter any tags that you applied to your original Ops Manager. For example, if you used the `pcf-opsmanager` tag to apply the firewall rule you created in Create Firewall Rules for the Network, then apply the same tag to this Ops Manager VM.
  - For **Internal IP**, select `Custom`. In the **Internal IP address** field, enter a spare address located within the reserved IP range configured in your existing Ops Manager Director. Do not use `10.0.0.1`, which is configured for the Gateway.

- For **External IP**, select **New static IP address…**. In the next form, enter a name for the static IP. For example, `om-public-ip`. Click **Reserve**. In the **External IP** drop-down, select the static IP address you just reserved.



4. Click **Create** to deploy the new Ops Manager VM. This may take a few moments.

5. Navigate to your DNS provider, and modify the entry that points a fully qualified domain name (FQDN) to the Ops Manager VM. Replace the original Ops Manager static IP address with the public IP address of the new Ops Manager VM you created in a previous step.

> **Note**: In order to set up Ops Manager authentication correctly, Pivotal recommends using a Fully Qualified Domain Name (FQDN) to access Ops Manager. Using an ephemeral IP address to access Ops Manager can cause authentication errors upon subsequent access.

What to Do Next

After you complete this procedure, continue the upgrade instructions in Upgrading Pivotal Cloud Foundry topic.

Later on, if you need to SSH into the Ops Manager VM to perform diagnostic troubleshooting, see SSH into Ops Manager.

Pivotal

PCF on OpenStack Requirements

Page last updated:

This guide describes how to install Pivotal Cloud Foundry (PCF) on OpenStack.

Supported Versions

PCF is supported on the OpenStack Liberty, Mitaka, and Newton releases. OpenStack is a collection of interoperable components and requires general OpenStack expertise to troubleshoot issues that may occur when installing Pivotal Cloud Foundry on particular releases and distributions.

In addition, to verify that your OpenStack platform is compatible with PCF, you can use the OpenStack Validator Tool.

General Requirements

The following are general requirements for deploying and managing a PCF deployment with Ops Manager and Pivotal Application Service (PAS):

- A wildcard DNS record that points to your router or load balancer. Alternatively, you can use a service such as xip.io. For example, `203.0.113.0.xip.io`.

  PAS gives each application its own hostname in your app domain. With a wildcard DNS record, every hostname in your domain resolves to the IP address of your router or load balancer, and you do not need to configure an A record for each app hostname. For example, if you create a DNS record `*.example.com` pointing to your load balancer or router, every application deployed to the `example.com` domain resolves to the IP address of your router.
- At least one wildcard TLS certificate that matches the DNS record you set up above, `*.example.com`.
- Sufficient IP allocation:
  - One static IP address for either HAProxy or one of your gorouters
  - One IP address for each VM instance
  - An additional IP address for each compilation worker
    So the formula for total IPs needed is `IPs needed = static IPs + VM instances + compilation workers`

  > **Note**: Pivotal recommends that you allocate at least 36 dynamic IP addresses when deploying Ops Manager and PAS. BOSH requires additional dynamic IP addresses during installation to compile and deploy VMs, install PAS, and connect to services.

- One or more NTP servers if not already provided by your IaaS.
- (**Recommended**) A network without DHCP available for deploying the PAS VMs.

  > **Note**: If you have DHCP, refer to the Troubleshooting Guide to avoid issues with your installation.

- (**Optional**) External storage. When you deploy PCF, you can select internal file storage or external file storage, either network-accessible or IaaS-provided, as an option in the PAS tile. Pivotal recommends using external storage whenever possible. See Upgrade Considerations for Selecting File Storage in Pivotal Cloud Foundry for a discussion of how file storage location affects platform performance and stability during upgrades.
- (**Optional**) External databases. When you deploy PCF, you can select internal or external databases for the BOSH Director and for PAS. Pivotal recommends using external databases in production deployments.
- (**Optional**) External user stores. When you deploy PCF, you can select a SAML user store for Ops Manager or a SAML or LDAP user store for PAS, to integrate existing user accounts.
- The most recent version of the Cloud Foundry Command Line Interface (cf CLI).

OpenStack Requirements

To deploy Pivotal Cloud Foundry on OpenStack, you must have a dedicated OpenStack project (formerly known as an OpenStack tenant) that meets the requirements described in this section.

- You must have Keystone access to the OpenStack tenant, including the following:
  - Auth URL
  - Username and password

- Project name
- Region (with multiple availability zones if you require high availability)
- SSL certificate for your wildcard domain (see below)

- You must have the ability to do the following:

  - Create and modify VM flavors. See the VM flavor configuration table
  - Enable DHCP if required
  - Create a network and then connect that network with a router to an external network
  - Create an external network with a pool of floating IP addresses
  - Boot VMs directly from image
  - Create two wildcard domains for separate system and app domains

- Your OpenStack project must have the following resources before you install PCF:

  - 118 GB of RAM
  - 22 available instances
  - 16 small VMs (1 vCPU, 1024 MB of RAM, 10 GB of root disk)
  - 3 large VMs (4 vCPU, 16384 MB of RAM, 10 GB of root disk)
  - 3 extra-large VMs (8 vCPU, 16 GB of RAM, 160 GB of ephemeral disk)
  - 56 vCPUs
  - 1 TB of storage
  - Nova or Neutron networking with floating IP support

By default, Pivotal Application Service (PAS) deploys the number of VM instances required to run a highly available configuration of PCF. If you are deploying a test or sandbox PCF that does not require HA, then you can scale down the number of instances in your deployment. For information about the number of instances required to run a minimal, non-HA PCF deployment, see Scaling PAS.

- Requirements for your Cinder backend:

  - PCF requires RAW root disk images. The Cinder backend for your OpenStack project must support RAW.
  - Pivotal recommends that you use a Cinder backend that supports snapshots. This is required for some BOSH functionalities.
  - Pivotal recommends enabling your Cinder backend to delete block storage asynchronously. If this is not possible, it must be able to delete multiple 20GB volumes within 300 seconds.

- Using an Overlay Network with VXLAN or GRE Protocols:

  - If an overlay network is being used with VXLAN or GRE protocols, the MTU of the created VMs must be adjusted to the best practices recommended by the plugin vendor (if any).
  - DHCP must be enabled in the internal network for the MTU to be assigned to the VMs automatically.
  - Review the Installing PAS on OpenStack topic to adjust your MTU values.
  - Failure to configure your overlay network correctly could cause Apps Manager to fail since applications will not be able to connect to the UAA.

> **Note**: If you are using IPsec, your resource usage will increase by approximately 36 bytes. View the Installing IPsec topic for information, including setting correct MTU values.

- Miscellaneous

  - Pivotal recommends granting complete access to the OpenStack logs to the operator managing the installation process.
  - Your OpenStack environment should be thoroughly tested and considered stable before deploying PCF. To validate that your OpenStack platform meets the needs of PCF, you can use the OpenStack Validator Tool.

OpenStack Permissions Guidelines

Pivotal recommends following the principle of least privilege by scoping privileges to the most restrictive permissions possible for a given role. See IaaS Permissions Guidelines for recommendations on how to create and scope OpenStack accounts for PCF.

OpenStack VM Flavors

Configure your OpenStack VM flavors as follows:

> Do not change the names of the VM flavors in the table below.

ID	Name	Memory_MB	Disk	Ephemeral	VCPUs

1	m1.small	2048	20	0	1
2	m1.medium	4096	40	0	2
3	m1.large	8192	80	0	4
4	m1.xlarge	16384	160	0	8

OpenStack Security Documents

- OpenStack credential configuration
- OpenStack credential creation
- OpenStack deployment configuration

These documents provide a general reference for OpenStack service credential management.

Install PCF on OpenStack

Complete the following procedures to install PCF on OpenStack:

1. Provisioning the OpenStack Infrastructure
2. Configuring Ops Manager Director after Deploying PCF on OpenStack
3. (Optional) Installing the PCF IPsec Add-On
4. Installing PAS after Deploying PCF on OpenStack

Provisioning the OpenStack Infrastructure

Page last updated:

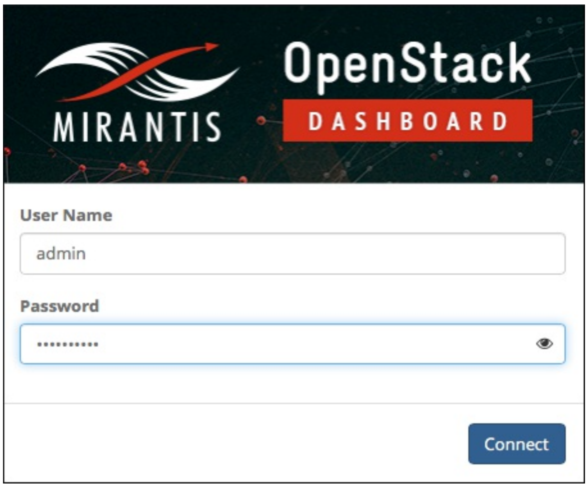
This guide describes how to provision the OpenStack infrastructure where you need to install Pivotal Cloud Foundry. Use this topic when Installing Pivotal Cloud Foundry on OpenStack.

After completing this procedure, complete all of the steps in the Configuring Ops Manager Director after Deploying PCF on OpenStack and Installing PAS after Deploying PCF on OpenStack topics.

Note: This document uses Mirantis OpenStack for screenshots and examples. The screens of your OpenStack vendor configuration interface may differ.

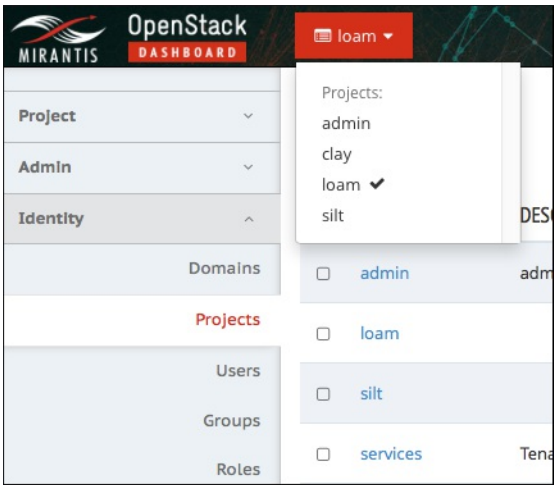
Step 1: Log in to the OpenStack Horizon Dashboard

1. Log in to the OpenStack Horizon dashboard.



2. Click **Connect**.

3. From the OpenStack project list dropdown, set the active project by selecting the project where you will deploy PCF.



Step 2: Configure Security

Warning: If you are using OpenStack Liberty or Mitaka, do not create the key pair with the OpenStack Horizon dashboard. Instead make sure that you generate the SSH key pair manually. For example, use the `ssh-keygen` command. Then follow the procedure below to import that key pair into

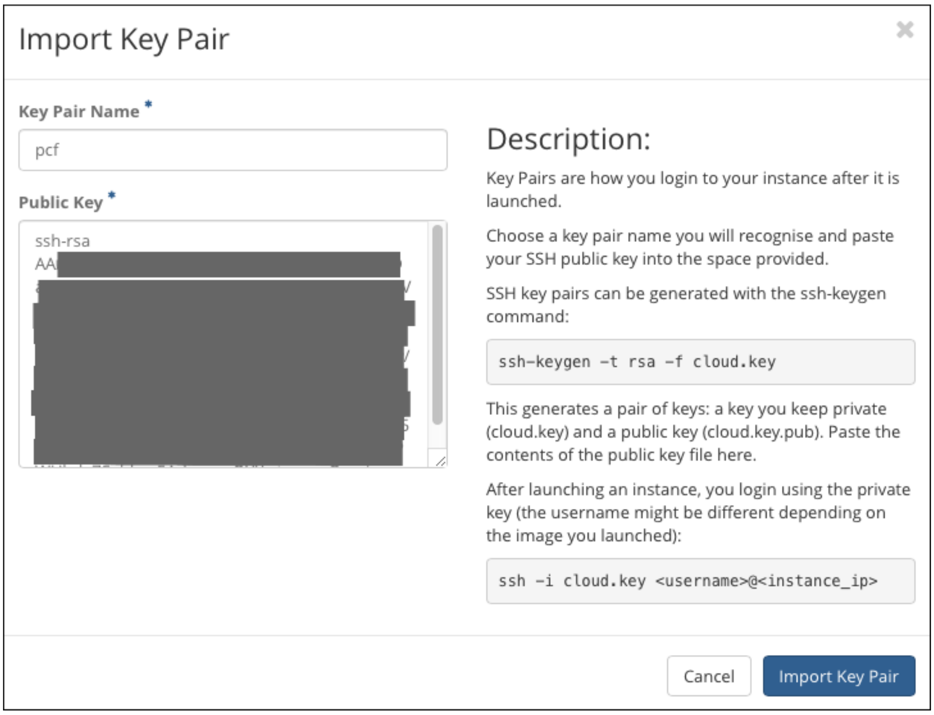
OpenStack. This is due to an OpenStack bug.

1. In the left navigation of your OpenStack Horizon dashboard, click **Project > Compute > Access & Security**.

2. Select the **Key Pairs** tab on the **Access & Security** page.

3. Click **Import Key Pair**.

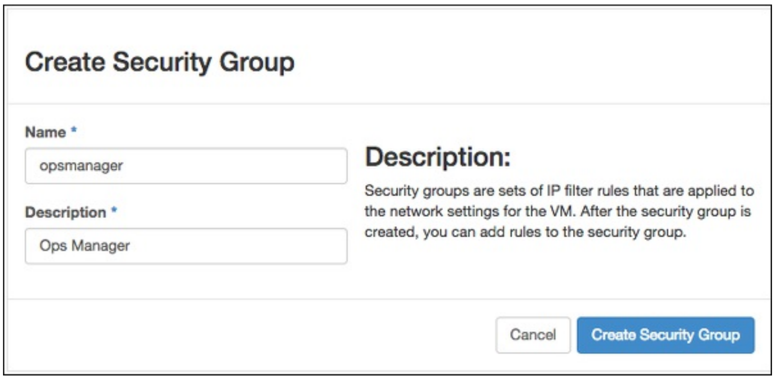
4. Enter a **Key Pair Name** and the contents of your public key in the **Public Key** field.



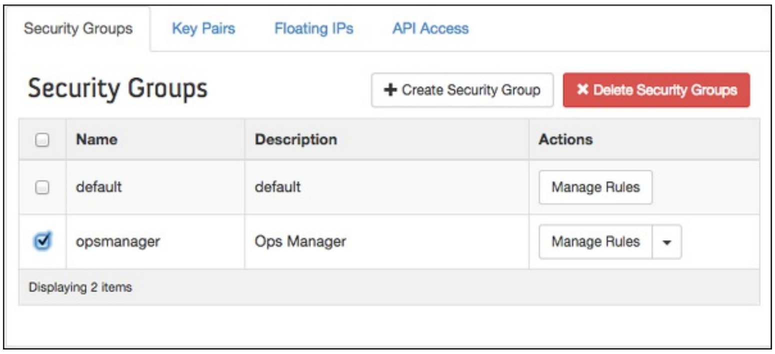
5. Click **Import Key Pair**.

6. In the left navigation, click **Access & Security** to refresh the page. The new key pair appears in the list.

7. Select the **Security Groups** tab. Click **Create Security Group** and create a group with the following properties:
   - **Name:** `opsmanager`
   - **Description:** `Ops Manager`



8. Select the checkbox for the `opsmanager` Security Group and click **Manage Rules**.

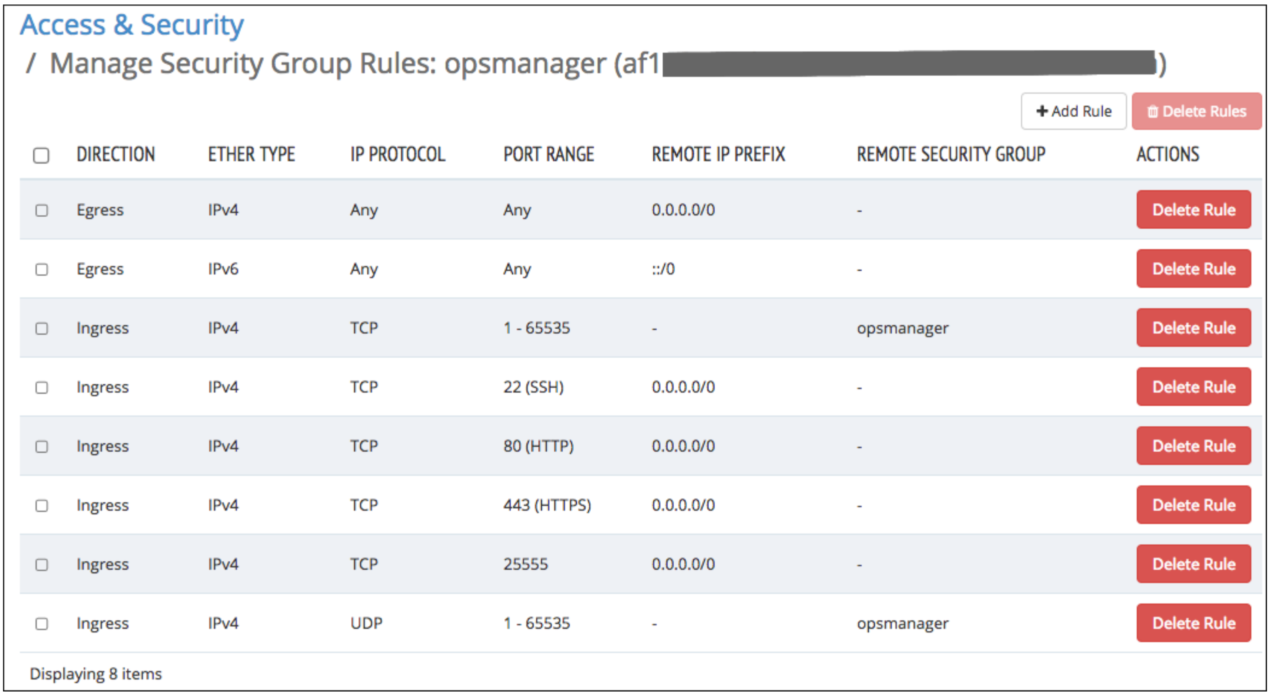


9. Add the following ingress access rules for HTTP, HTTPS, and SSH as shown in the table below. The rules with `opsmanager` in the Remote column have restricted access to that particular Security Group.

> **Note:** Adjust the remote sources as necessary for your own security compliance. Pivotal recommends limiting remote access to Ops Manager to IP ranges within your organization.

Direction	Ether Type	IP Protocol	Port/Port Range	Remote
Ingress	IPv4	TCP	22 (SSH)	0.0.0.0/0 (CIDR)
Ingress	IPv4	TCP	80 (HTTP)	0.0.0.0/0 (CIDR)
Ingress	IPv4	TCP	443 (HTTPS)	0.0.0.0/0 (CIDR)
Ingress	IPv4	TCP	25555	0.0.0.0/0 (CIDR)
Ingress	IPv4	TCP	1-65535	opsmanager
Ingress	IPv4	UDP	1-65535	opsmanager

10. Leave the existing default egress access rules as shown in the screenshot below.



Step 3: (Optional) Run the CF OpenStack Validator Tool

As an optional but recommended step, you can now run the CF OpenStack Validator tool against your OpenStack tenant to verify support for PCF.

1. Follow the directions for running the CF OpenStack Validator Tool.

2. When configuring the CPI version used by the Validator, specify the OpenStack CPI version indicated in the PCF Ops Manager Release Notes for the

PCF release that you are planning to deploy.

Troubleshooting the output of the CF OpenStack Validator tool is beyond the scope of this document.

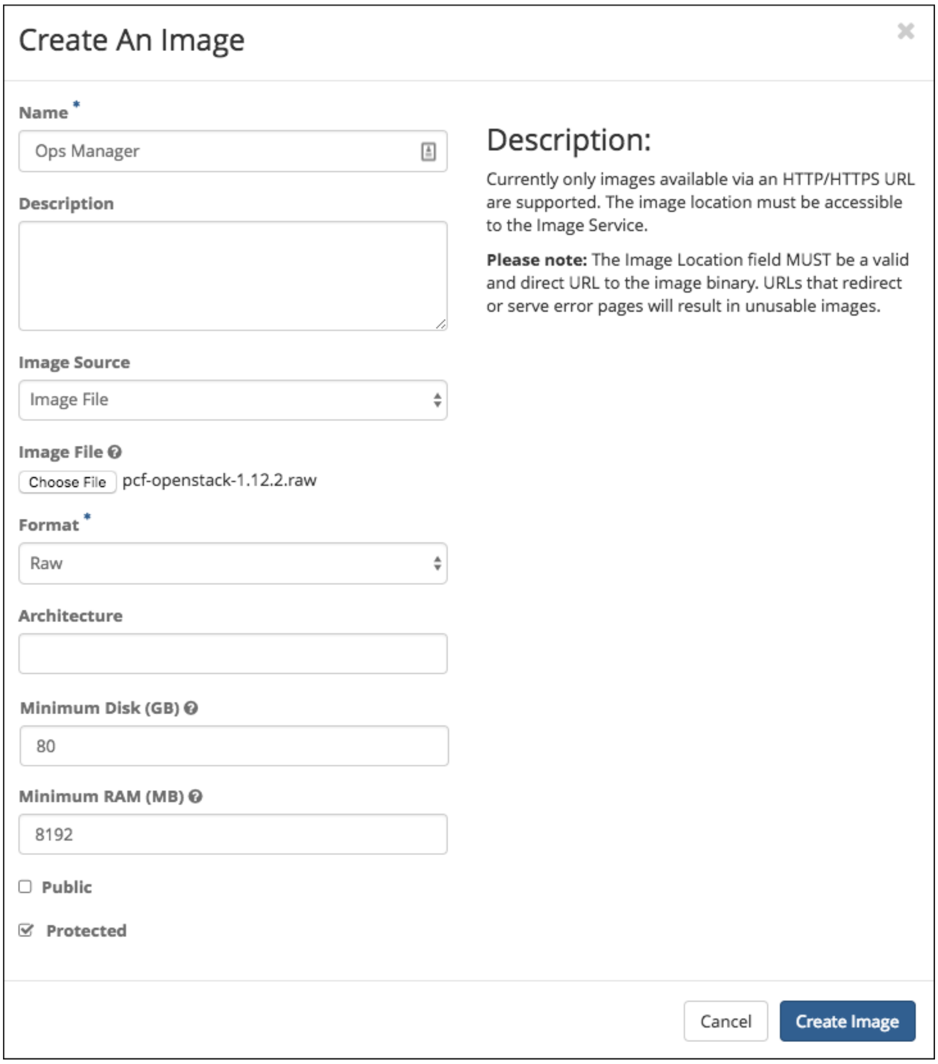
Step 4: Create Ops Manager Image

You can create the Ops Manager image in OpenStack using the OpenStack Horizon dashboard.

> **Note**: If your Horizon Dashboard does not support file uploads, you must use the Glance CLI client.

To create an Ops Manager image in OpenStack, perform the following steps:

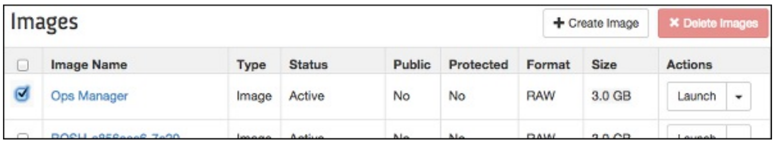
1. Download the **Pivotal Cloud Foundry Ops Manager for OpenStack** image file from Pivotal Network.
2. In the left navigation of your OpenStack dashboard, click **Project > Compute > Images**.
3. Click **Create Image**. Complete the **Create An Image** page with the following information:
    - **Name**: Enter `Ops Manager`.
    - **Image Source**: Select **Image File**.
    - **Image File**: Click **Choose File**. Browse to and select the image file that you downloaded from Pivotal Network.
    - **Format**: Select **Raw**.
    - **Minimum Disk (GB)**: Enter `80`.
    - **Minimum RAM (MB)**: Enter `8192`.
    - Deselect the **Public** checkbox.
    - Select the **Protected** checkbox.



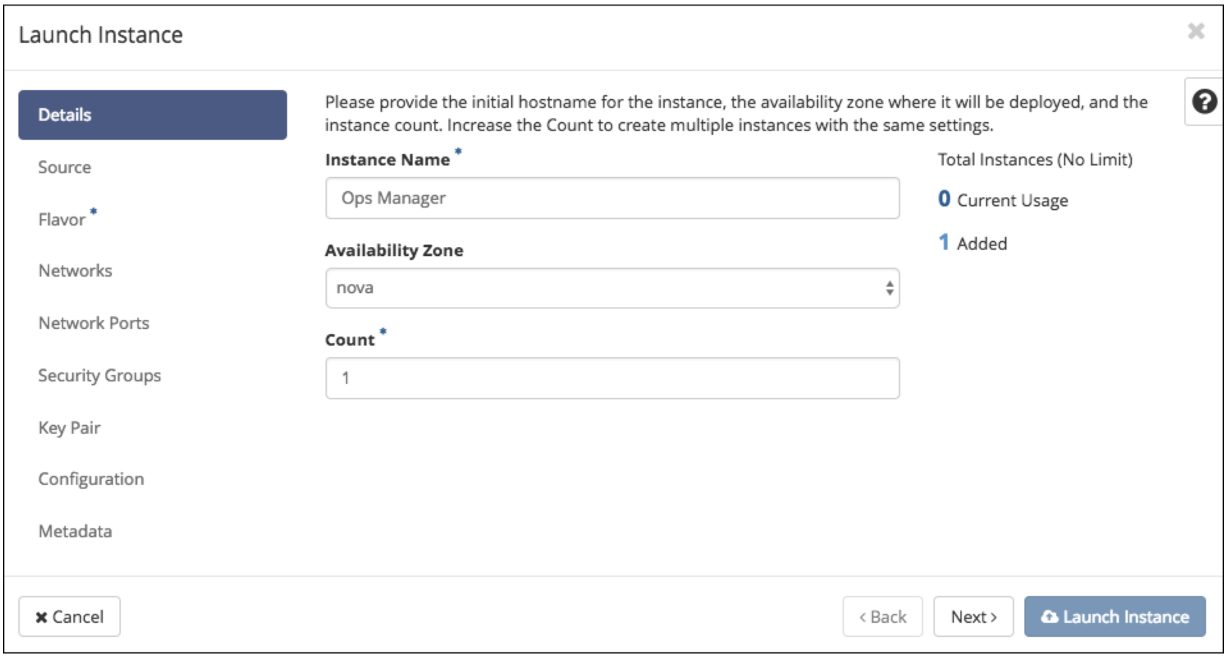
4. Click **Create Image**.

Step 5: Launch Ops Manager VM

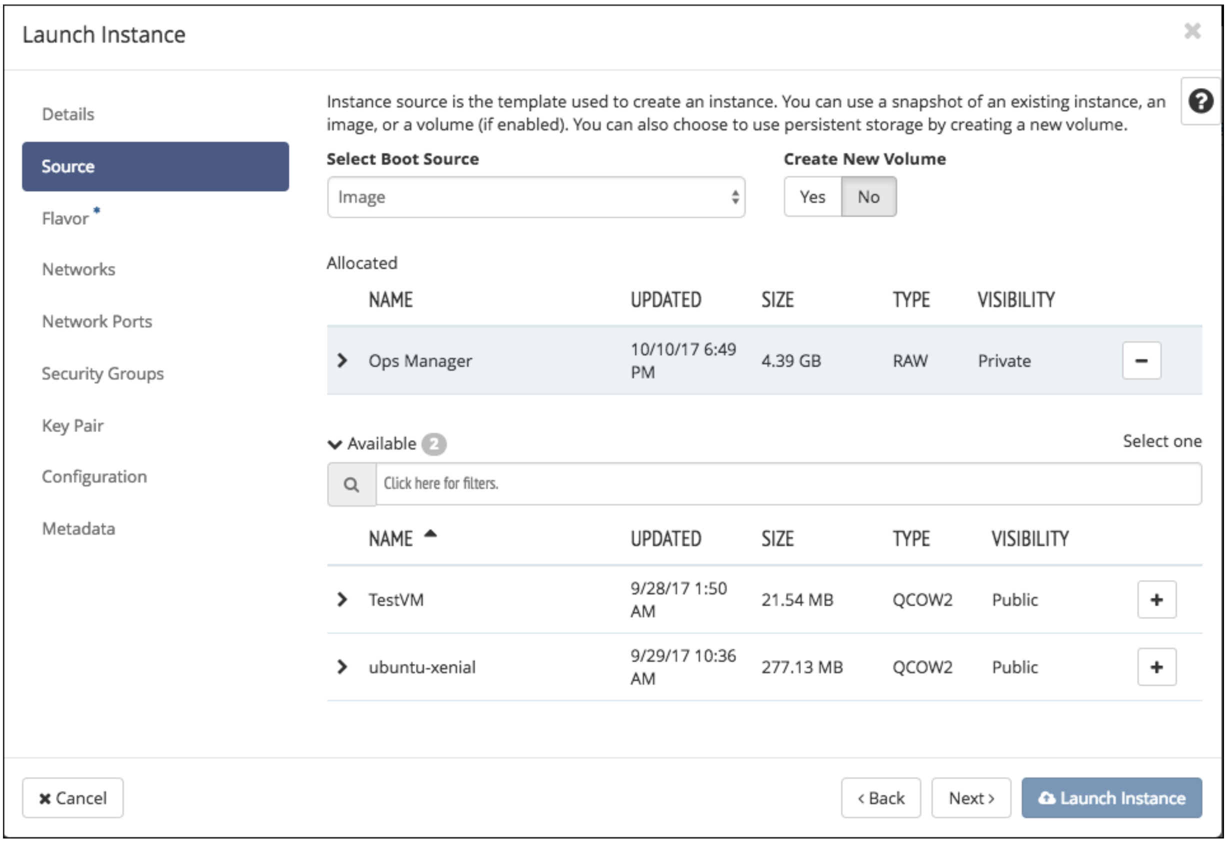
1. In the left navigation of your OpenStack dashboard, click **Project > Compute > Images**.
2. Click **Launch**.



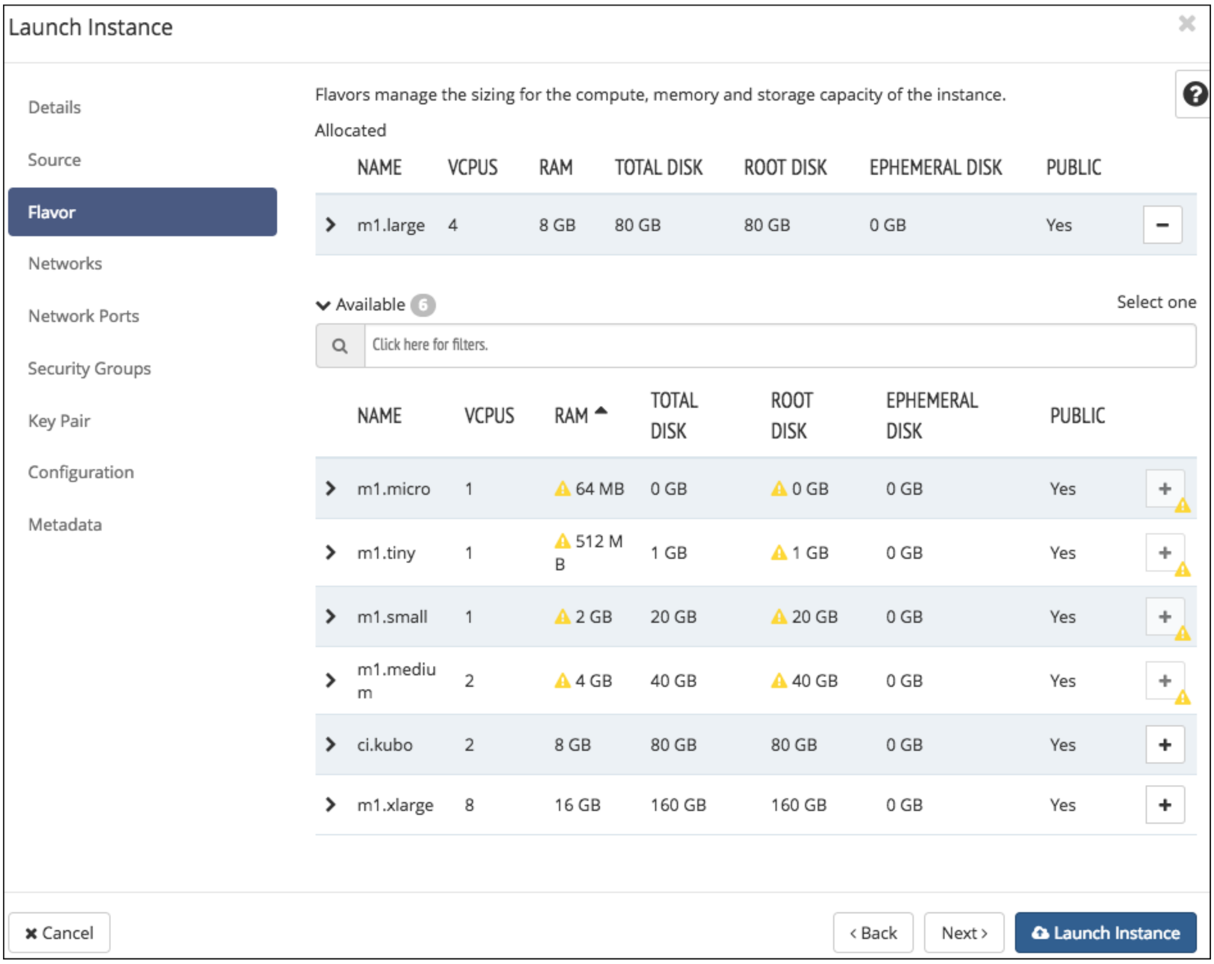
3. In the **Details** tab, specify the following values:
   - **Instance Name**: Enter `Ops Manager`.
   - **Availability Zone**: Use the drop-down menu to select an availability zone. You specify this availability zone in the [Complete the Availability Zones Pages]() step of [Configuring Ops Manager Director]().
   - **Count**: Do not change from the default value of **1**.



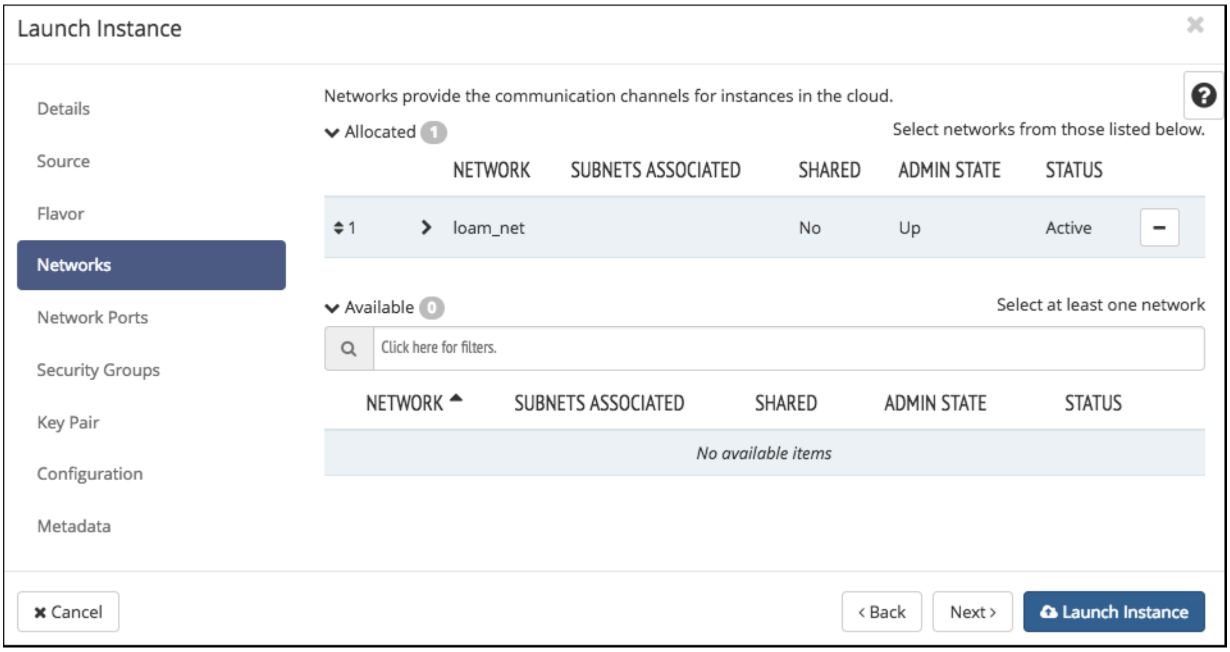
4. In the **Source** tab, specify the following values:
   - **Select Boot Source**: Select **Image**.
   - **Create New Volume**: Leave **No** selected.
   - **Allocated**: Make sure **Ops Manager** is selected.



5. In the **Flavor** tab, click the plus button for **m1.large**.

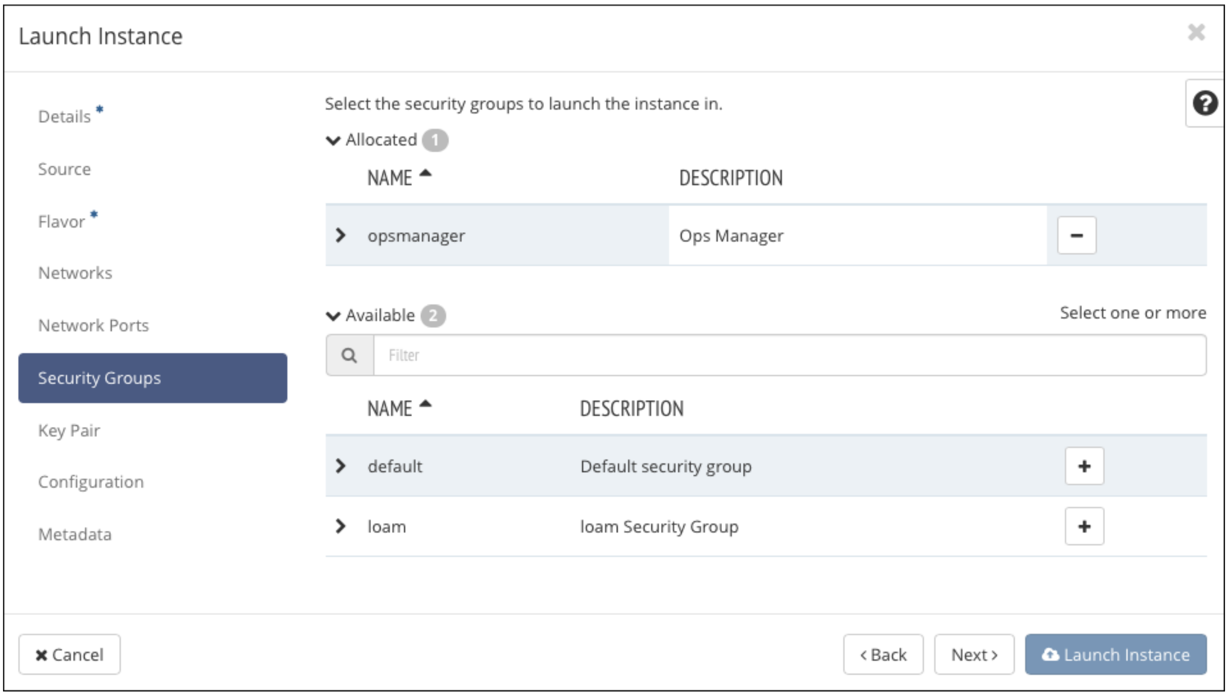


6. In the **Networks** tab, select a private subnet. You add a Floating IP to this network in a later step.

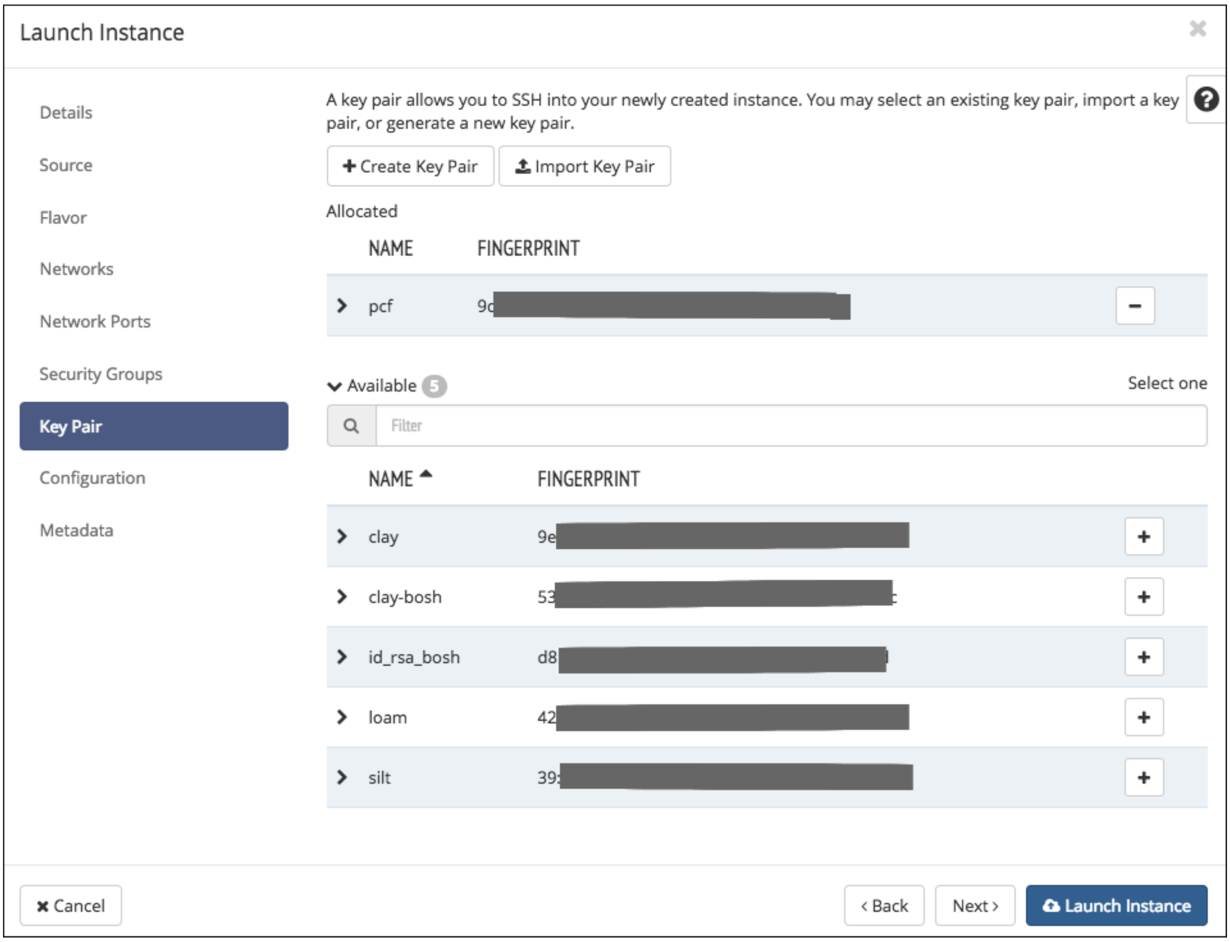


7. Skip the **Network Ports** tab.

8. In the **Security Groups** tab, select the **opsmanager** security group that you created in Step 2: Configure Security. Deselect all other Security Groups.



9. In the **Key Pair** tab, select the key pair that you imported in Step 2: Configure Security.



10. Skip the **Configuration** and **Metadata** tabs.

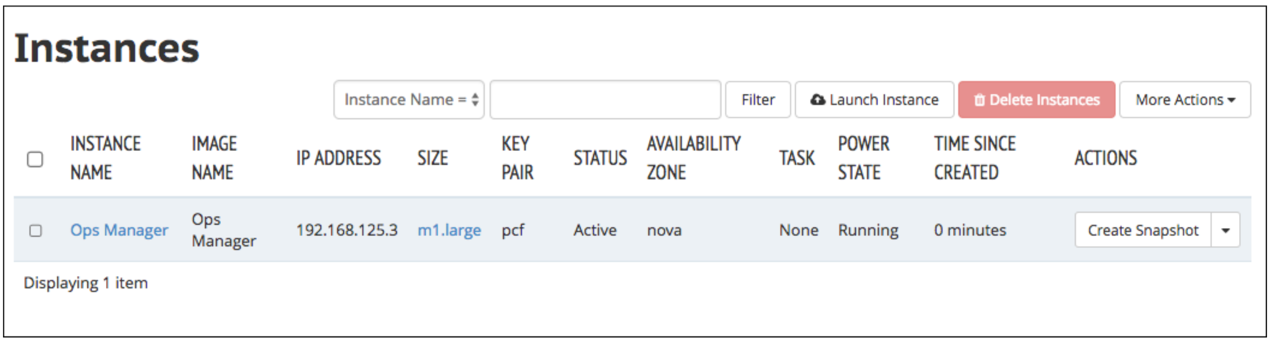
11. Click **Launch Instance**. This step starts your new Ops Manager instance.

Step 6: Associate a Floating IP Address

1. In the left navigation of your OpenStack dashboard, click **Project > Compute > Instances**.

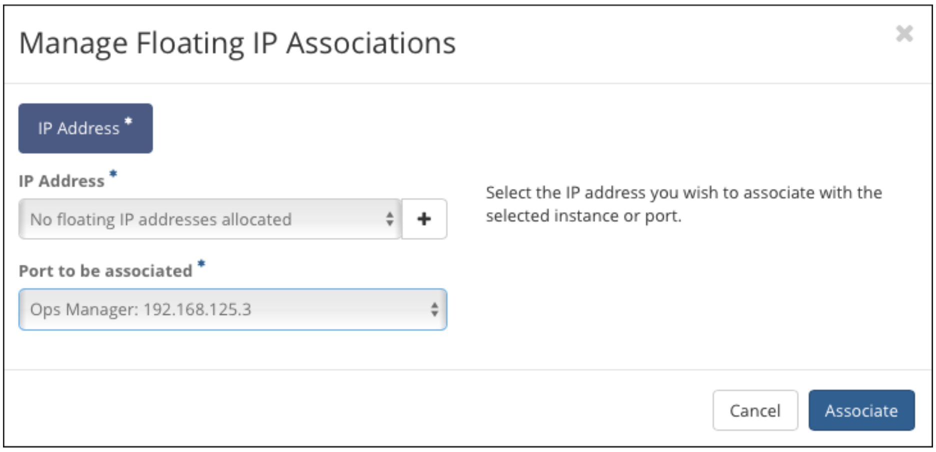
2. Wait until the **Power State** of the Ops Manager instance shows as *Running*.

3. Record the private **IP Address** of the Ops Manager instance.



You must provide this IP Address when you perform Step 6: Complete the Create Networks Page in Ops Manager.

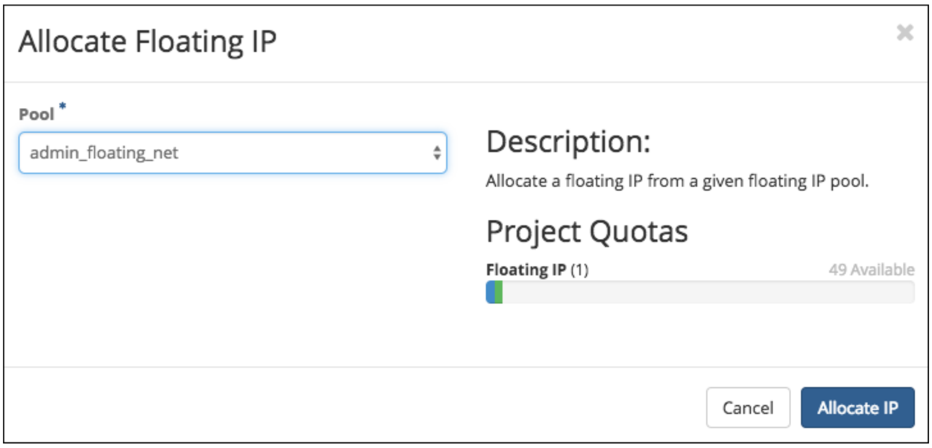
4. Select the **Ops Manager** checkbox. Click the **Actions** drop-down menu and select **Associate Floating IP**. The **Manage Floating IP Associations**



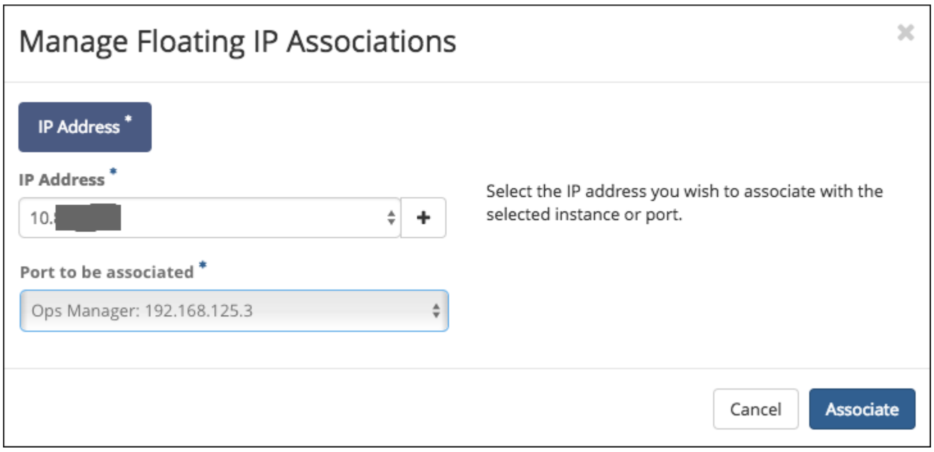
screen appears.

5. Under **IP Address**, click the plus button (**+**). The **Allocate Floating IP** screen appears.

6. Under **Pool**, select an IP Pool and click **Allocate IP**.



7. Under **Port to be associated**, select your **Ops Manager** instance.

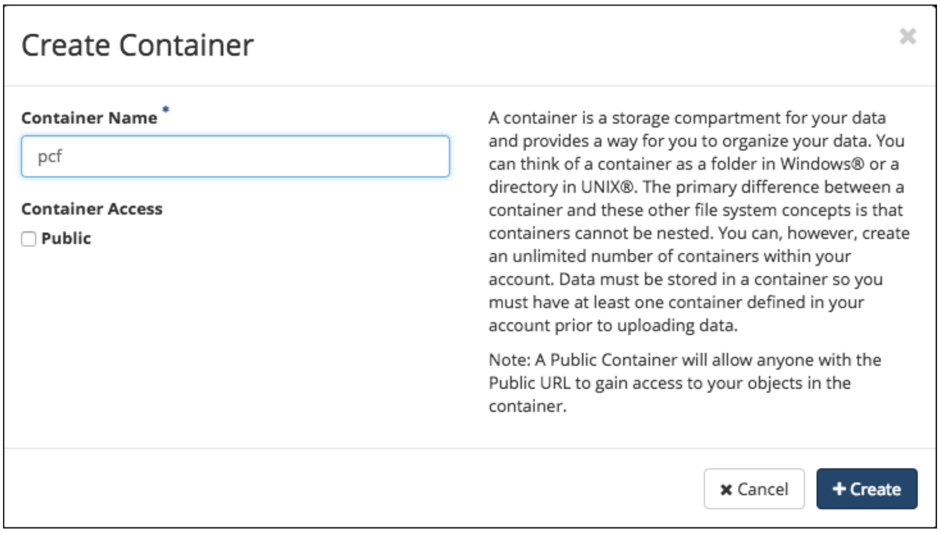


8. Click **Associate**.

Step 7: Add Blob Storage

1. In the left navigation of your OpenStack dashboard, click **Project > Object Store > Containers**.

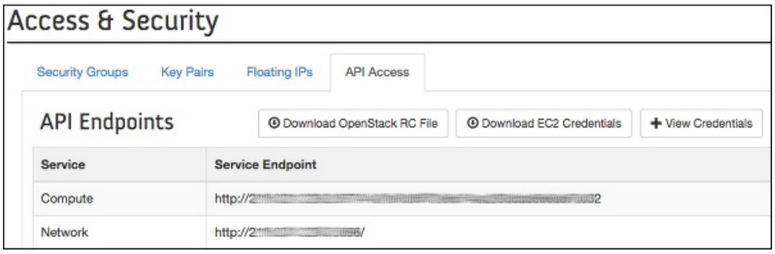
2. Click **Create Container**. Create a container with the following properties:
   - **Container Name**: Enter `pcf`.
   - **Container Access**: Leave **public** unselected.



3. Click **Create**.

Step 8: Download Credentials for S3 Blob Storage

1. In the left navigation of your OpenStack dashboard, click **Project** > **Compute** > **Access & Security**. Select the **API Access** tab.



2. Click **Download EC2 Credentials**.

3. Unzip the downloaded credentials.

4. If you select **S3 Compatible Blobstore** in your Ops Manager Director Config, you need the contents of this file to complete the configuration.

Step 9: Create a DNS Entry

> **Note**: For security, Ops Manager 1.7 and later require you to create a fully qualified domain name in order to access Ops Manager during the initial configuration.

Create a DNS entry for the floating IP address that you assigned to Ops Manager in Step 6: Associate a Floating IP Address.

You must use this fully qualified domain name when you log into Ops Manager for the first time.

Step 10: Configure Ops Manager Director for OpenStack

After completing this procedure, complete all of the steps in the Configuring Ops Manager Director after Deploying PCF on OpenStack and Installing PAS after Deploying PCF on OpenStack topics.

Return to Installing Pivotal Cloud Foundry on OpenStack.

Configuring Ops Manager Director for OpenStack

Page last updated:

This topic describes how to configure the Ops Manager Director after deploying Pivotal Cloud Foundry (PCF) on OpenStack. Use this topic when Installing Pivotal Cloud Foundry on OpenStack.

Before beginning this procedure, ensure that you have successfully completed all steps in the Provisioning the OpenStack Infrastructure topic. After you complete this procedure, follow the instructions in the Installing PAS after Deploying PCF on OpenStack topic.

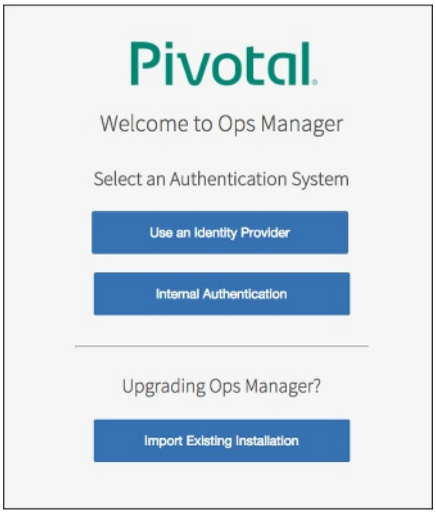
> **Note**: You can also perform the procedures in this topic using the Ops Manager API. For more information, see the Using the Ops Manager API topic.

Step 1: Access Ops Manager

> **Note**: For security, Ops Manager 1.7 and later require that you log in using a fully qualified domain name to access Ops Manager.

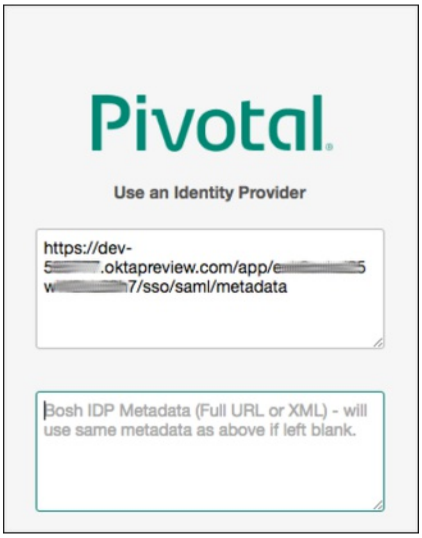
1. In a web browser, navigate to the fully qualified domain you created in the Create a DNS Entry step of Provisioning the OpenStack Infrastructure.

2. When Ops Manager starts for the first time, you must choose one of the following:
    - Use an Identity Provider: If you use an Identity Provider, an external identity server maintains your user database.
    - Internal Authentication: If you use Internal Authentication, Pivotal Cloud Foundry (PCF) maintains your user database.



Use an Identity Provider

1. Log in to your IdP console and download the IdP metadata XML. Optionally, if your IdP supports metadata URL, you can copy the metadata URL instead of the XML.



2. Copy the IdP metadata XML or URL to the Ops Manager **Use an Identity Provider** log in page.

> **Note**: The same IdP metadata URL or XML is applied for the BOSH Director. If you use a separate IdP for BOSH, copy the metadata XML or URL from that IdP and enter it into the BOSH IdP Metadata text box in the Ops Manager log in page.

3. Enter your **Decryption passphrase**. Read the **End User License Agreement**, and select the checkbox to accept the terms.

4. Your Ops Manager log in page appears. Enter your username and password. Click **Login**.

5. Download your SAML Service Provider metadata (SAML Relying Party metadata) by navigating to the following URLs:
   - **5a.** Ops Manager SAML service provider metadata: `https://OPS-MAN-FQDN:443/uaa/saml/metadata`
   - **5b.** BOSH Director SAML service provider metadata: `https://BOSH-IP-ADDRESS:8443/saml/metadata`

> **Note**: To retrieve your `BOSH-IP-ADDRESS`, navigate to the **Ops Manager Director** tile > **Status** tab. Record the **Ops Manager Director** IP address.

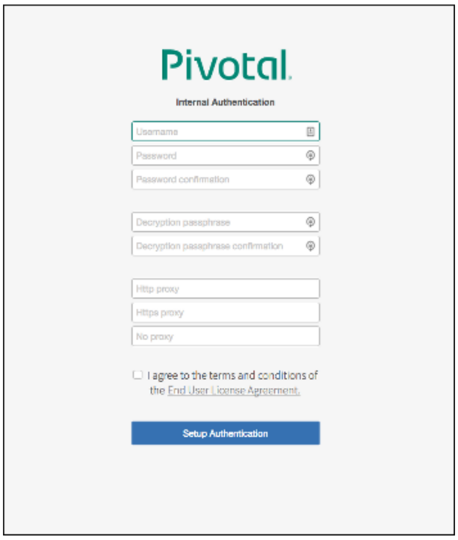
6. Configure your IdP with your SAML Service Provider metadata. Import the Ops Manager SAML provider metadata from Step 5a above to your IdP. If your IdP does not support importing, provide the values below.
   - **Single sign on** URL: `https://OPS-MAN-FQDN:443/uaa/saml/SSO/alias/OPS-MAN-FQDN`
   - **Audience URI** (SP Entity ID): `https://OP-MAN-FQDN:443/uaa`
   - **Name ID**: Email Address
   - SAML authentication requests are always signed

7. Import the BOSH Director SAML provider metadata from Step 5b to your IdP. If the IdP does not support an import, provide the values below.
   - **Single sign on** URL: `https://BOSH-IP:8443/saml/SSO/alias/BOSH-IP`
   - **Audience URI** (SP Entity ID): `https://BOSH-IP:8443`
   - **Name ID**: Email Address
   - SAML authentication requests are always signed

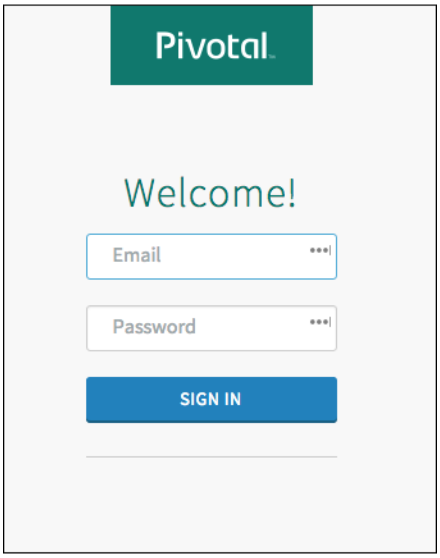
8. Return to the **Ops Manager Director** tile, and continue with the configuration steps below.

Use Internal Authentication

1. When redirected to the **Internal Authentication** page, you must complete the following steps:
   - Enter a **Username**, **Password**, and **Password confirmation** to create an Admin user.
   - Enter a **Decryption passphrase** and the **Decryption passphrase confirmation**. This passphrase encrypts the Ops Manager datastore, and is not recoverable.
   - If you are using an **HTTP proxy** or **HTTPS proxy**, follow the instructions in the Configuring Proxy Settings for the BOSH CPI topic.
   - Read the **End User License Agreement**, and select the checkbox to accept the terms.
   - Click **Setup Authentication**.



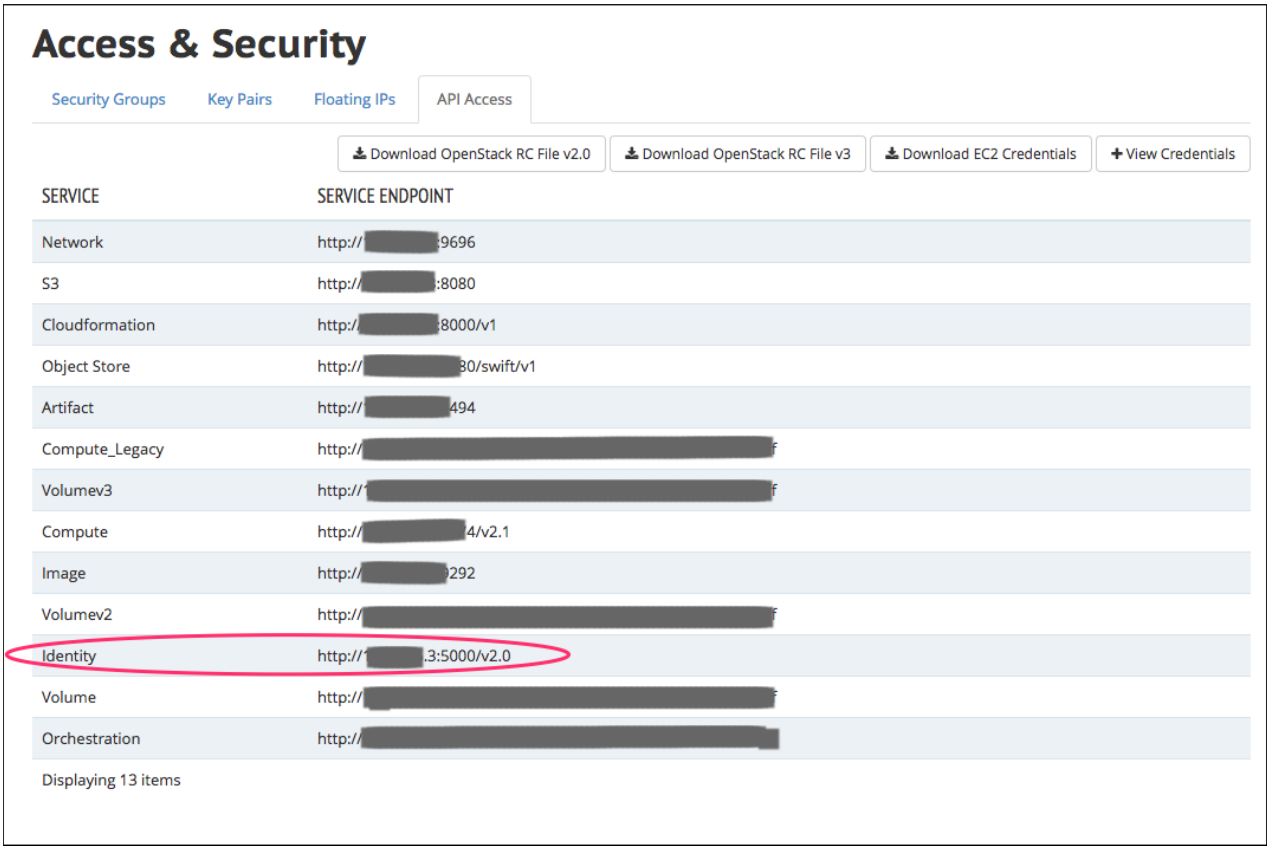
2. Log in to Ops Manager with the Admin username and password you created in the previous step.



Step 2: Complete the OpenStack Config Page

1. In the left navigation of your OpenStack dashboard, click **Project > Compute > Access & Security**. Select the **API Access** tab.

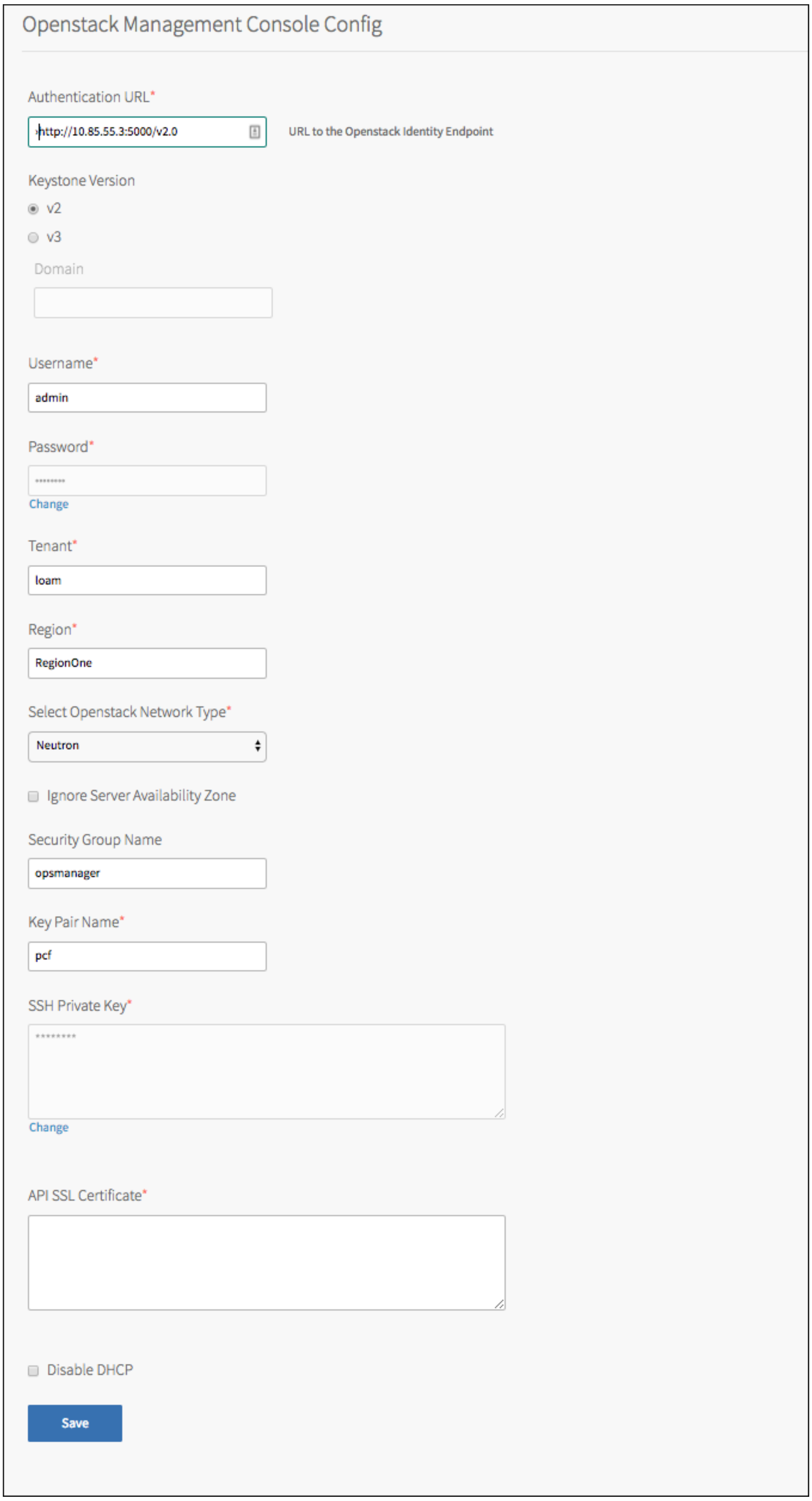
2. Record the **Service Endpoint** for the **Identity** service. You use this Service Endpoint as the Authentication URL for Ops Manager in a later step.



3. In the PCF Ops Manager Installation Dashboard, click the **Ops Manager Director** tile.

4. Select **OpenStack Config**.

5. Complete the **OpenStack Management Console Config** page with the following information:
   - **Authentication URL**: Enter the Service Endpoint for the Identity service that you recorded in a previous step.
   - **Keystone Version**: Choose a Keystone version. If you choose **v3**, you must enter a **Domain** to authenticate against.
   - **Username**: Enter your OpenStack Horizon username.
   - **Password**: Enter your OpenStack Horizon password.
   - **Tenant**: Enter your OpenStack tenant name.
   - **Region**: Enter `RegionOne`, or another region if recommended by your OpenStack administrator.
   - **Select OpenStack omNetwork Type**: Select either **Nova**, the legacy OpenStack networking model, or **Neutron**, the newer networking model.
   - **Ignore Server Availability Zone**: Do not select the checkbox.
   - **Security Group Name**: Enter `opsmanager`. You created this Security Group when Provisioning the OpenStack Infrastructure.
   - **Key Pair Name**: Enter the name of the key pair that you created in the Configure Security step of the Provisioning the OpenStack Infrastructure topic.
   - **SSH Private Key**: In a text editor, open the key pair file that you downloaded in the Configure Security step of the Provisioning the OpenStack Infrastructure topic. Copy and paste the contents of the key pair file into the field.
   - (Optional) **API SSL Certificate**: If, in your OpenStack Dashboard, you have configured API SSL termination, enter your **API SSL Certificate**.
   - **Disable DHCP**: Do not select the checkbox unless your setup requires it.
   - **Select OpenStack Network Type**: Select either **Nova**, the legacy networking model, or **Neutron**, the OpenStack networking model.

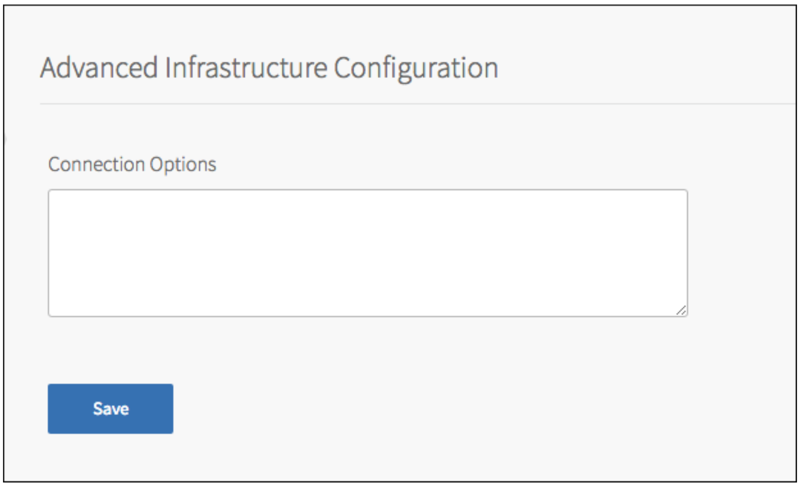


6. Click **Save**.

Step 3: (Optional) Complete the Advanced Config Page

> **Note:** This is an advanced option. Most users leave this field blank.

1. In Ops Manager, select **Advanced Infrastructure Config**.

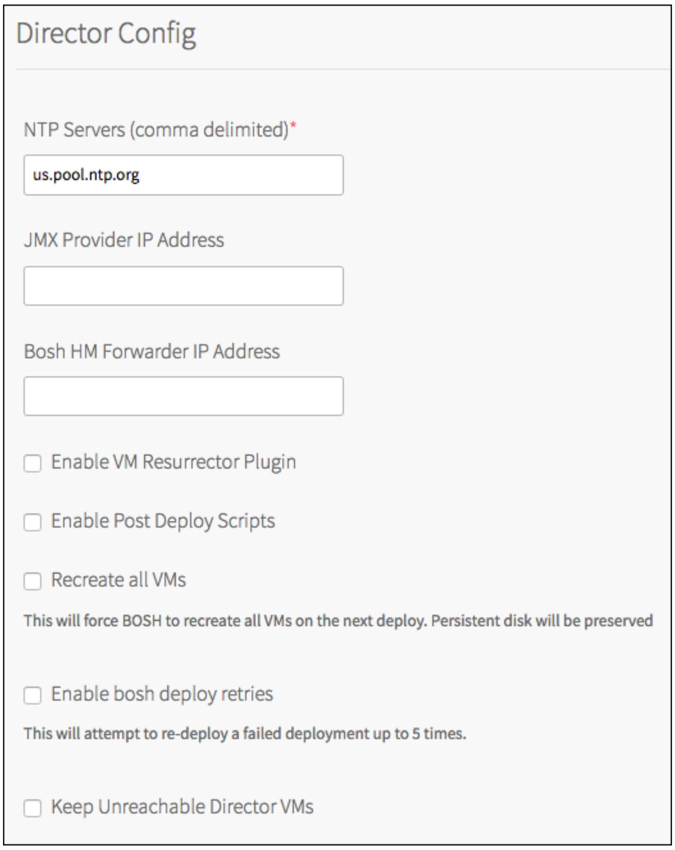


2. If your OpenStack environment requires specific connection options, enter them in the **Connection Options** field in JSON format. For example: `'connection_options' => { 'read_timeout' => 200 }`

3. Click **Save**.

Step 4: Complete the Director Config Page

1. In Ops Manager, select **Director Config**.



2. Enter one or more NTP servers in the **NTP Servers (comma delimited)** field. For example, `us.pool.ntp.org`.

3. Leave the **JMX Provider IP Address** field blank.

> **Note:** Starting from PCF v2.0, BOSH-reported system metrics are available in the Loggregator Firehose by default. Therefore, if you continue to use PCF JMX Bridge for consuming them outside of the Firehose, you may receive duplicate data. To prevent this, leave the **JMX Provider IP Address** field blank. For additional guidance, see BOSH System Metrics Available in Loggregator Firehose.

4. Leave the **Bosh HM Forwarder IP Address** field blank.

> **Note**: Starting from PCF v2.0, BOSH-reported system metrics are available in the Loggregator Firehose by default. Therefore, if you continue to use the BOSH HM Forwarder for consuming them, you may receive duplicate data. To prevent this, leave the **Bosh HM Forwarder IP Address** field blank. For additional guidance, see BOSH System Metrics Available in Loggregator Firehose.

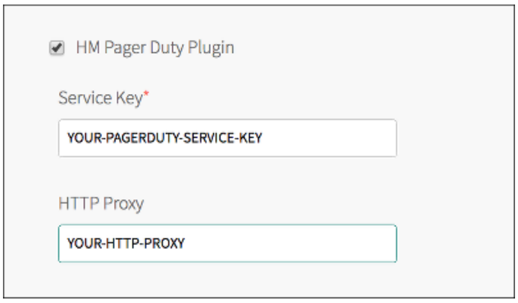
5. Select the **Enable VM Resurrector Plugin** checkbox to enable the Ops Manager Resurrector functionality and increase Pivotal Application Service availability.

6. Select **Enable Post Deploy Scripts** to run a post-deploy script after deployment. This script allows the job to execute additional commands against a deployment.

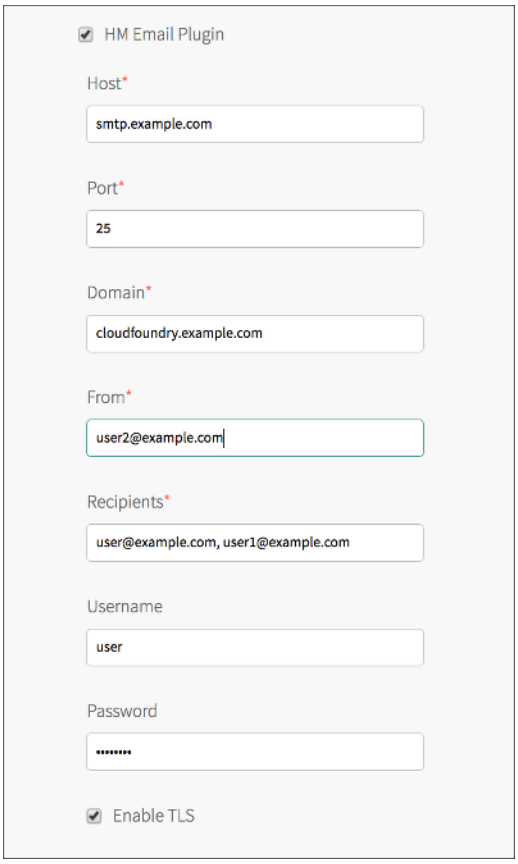
7. Select **Recreate all VMs** to force BOSH to recreate all VMs on the next deploy. This process does not destroy any persistent disk data.

8. Select **Enable bosh deploy retries** if you want Ops Manager to retry failed BOSH operations up to five times.

9. Select **Keep Unreachable Director VMs** if you want to preserve Ops Manager Director VMs after a failed deployment for troubleshooting purposes.



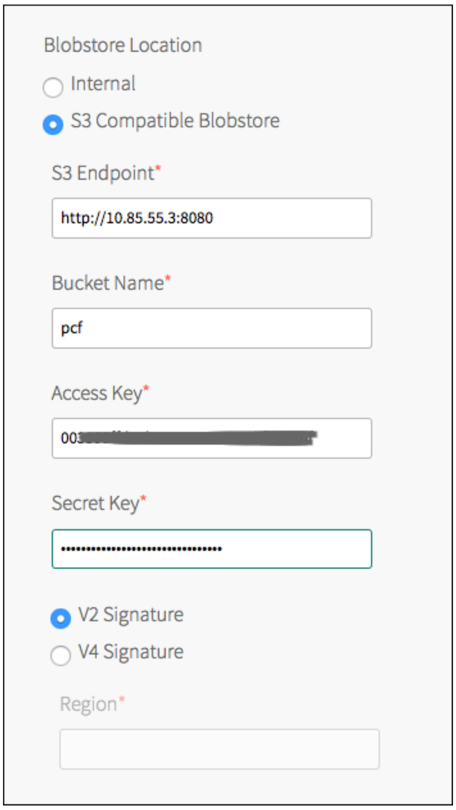
10. Select **HM Pager Duty Plugin** to enable Health Monitor integration with PagerDuty.
    - **Service Key**: Enter your API service key from PagerDuty.
    - **HTTP Proxy**: Enter an HTTP proxy for use with PagerDuty.



11. Select **HM Email Plugin** to enable Health Monitor integration with email.
    - **Host**: Enter your email hostname.
    - **Port**: Enter your email port number.
    - **Domain**: Enter your domain.
    - **From**: Enter the address for the sender.

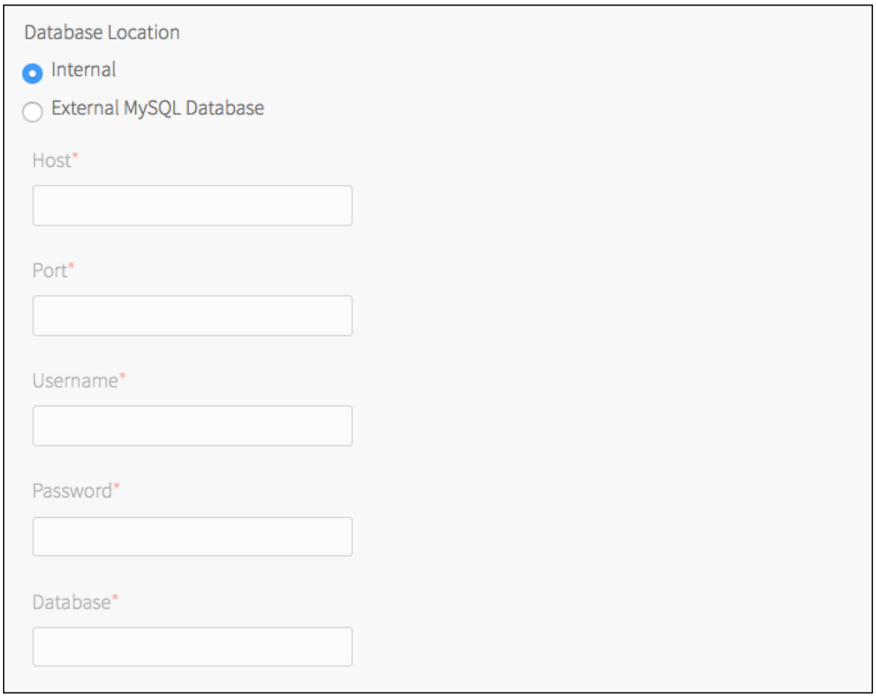
- **Recipients**: Enter comma-separated addresses of intended recipients.
- **Username**: Enter the username for your email server.
- **Password**: Enter the password password for your email server.
- **Enable TLS**: Select this checkbox to enable Transport Layer Security.

12. For **Blobstore Location**, select **S3 Compatible Blobstore** and complete the following steps using information from the `ec2rc.sh` file:



- In a text editor, open the `ec2rc.sh` file that you downloaded in [Step 8: Download Credentials for S3 Blob Storage](). Use the file to populate the fields described below.
- **Blobstore Location**: Select the **S3 Compatible Blobstore** option.
- **S3 Endpoint**: Use **S3_URL** from the `ec2rc.sh` file.
- **Bucket Name**: Enter `pcf`.
- **Access Key**: Use **EC2_ACCESS_KEY** from the `ec2rc.sh` file.
- **Secret Key**: Use **EC2_SECRET_KEY** from the `ec2rc.sh` file.

13. Select a **Database Location**. By default, PCF deploys and manages a database for you. If you choose to use an **External MySQL Database**, complete the associated fields with information obtained from your external MySQL Database provider.



14. (Optional) **Director Workers** sets the number of workers available to execute Director tasks. This field defaults to `5`.

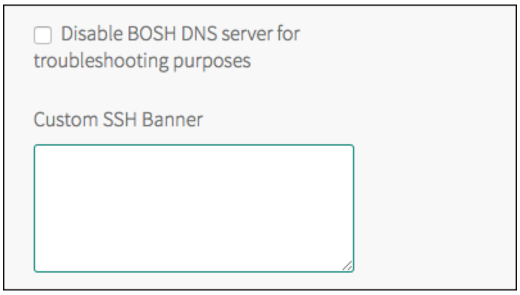
15. (Optional) For **Max Threads**, enter the number of operations the Ops Manager Director can perform simultaneously.

16. (Optional) To add a custom URL for your Ops Manager Director, enter a valid hostname in **Director Hostname**. You can also use this field to configure a load balancer in front of your Ops Manager Director.

17. (Optional) To disable BOSH DNS, select the **Disable BOSH DNS server for troubleshooting purposes** checkbox. For more information about the BOSH DNS service discovery mechanism, see BOSH DNS Service Discovery for Application Containers in the Pivotal Application Service (PAS) Release Notes.

    Breaking Change: Do not disable BOSH DNS without consulting Pivotal Support. For more information about disabling BOSH DNS, see Disabling or Opting Out of BOSH DNS in PCF in the Pivotal Knowledge Base.

18. (Optional) To set a custom banner that users see when logging in to the Director using SSH, enter text in the **Custom SSH Banner** field.

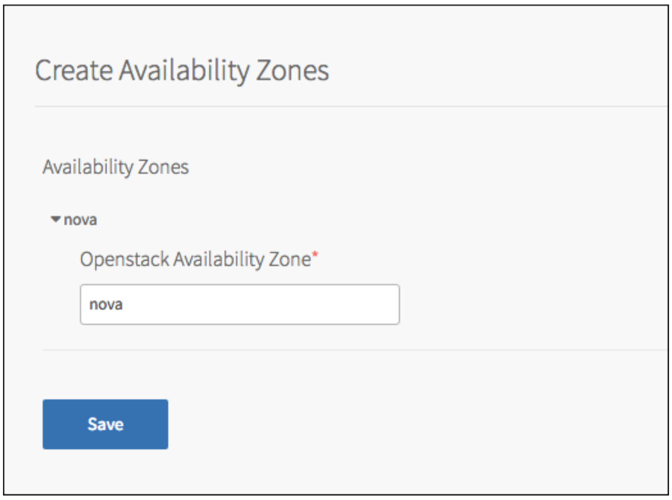


19. Click **Save**.

Note: If you select to use an internal database, back up your data frequently to ensure you have saved the latest copy.

Step 5: Complete the Create Availability Zones Page

1. In Ops Manager, select **Create Availability Zones**.



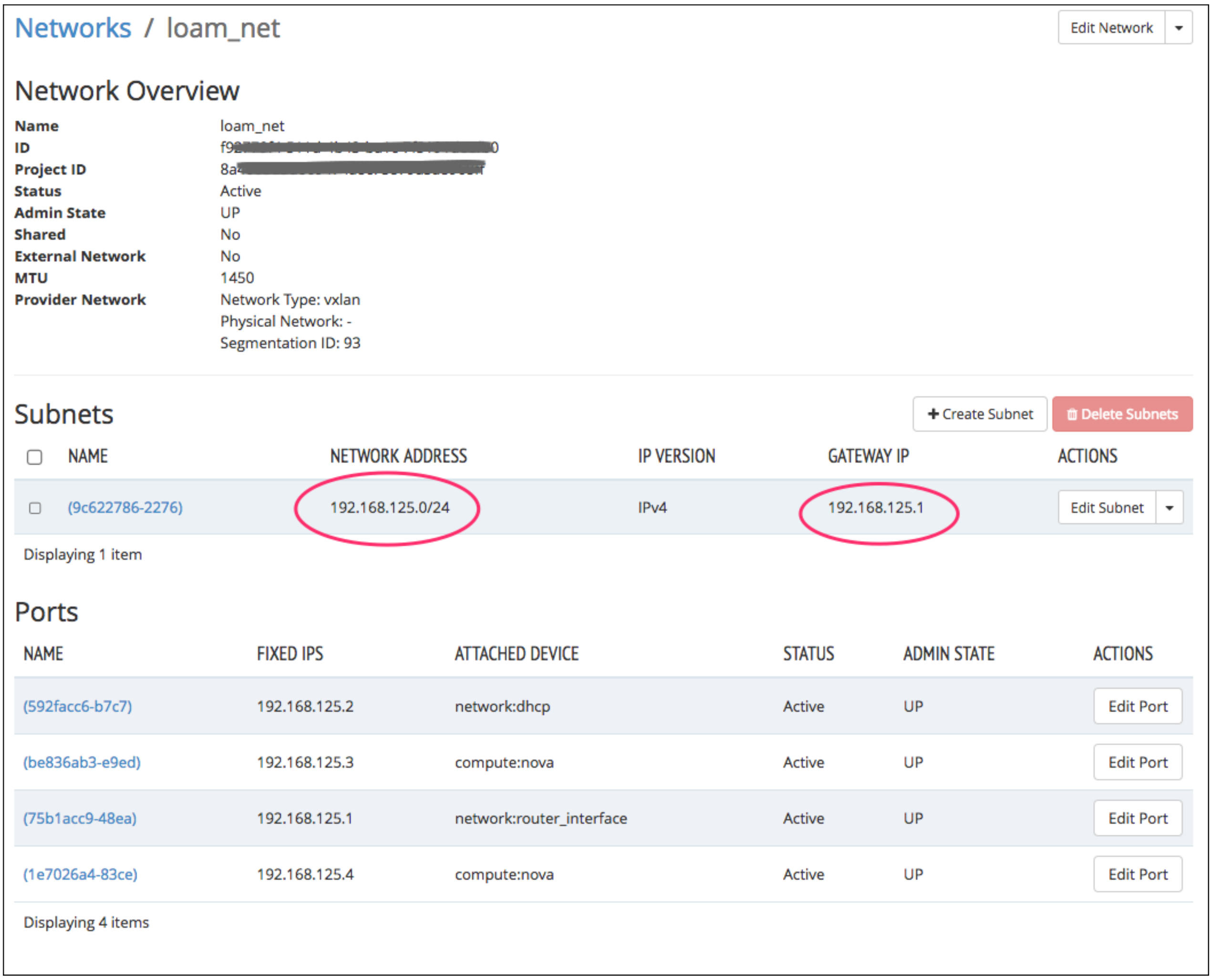
2. Enter the name of the availability zone that you selected when [Provisioning the OpenStack Infrastructure](#).

3. Click **Save**.

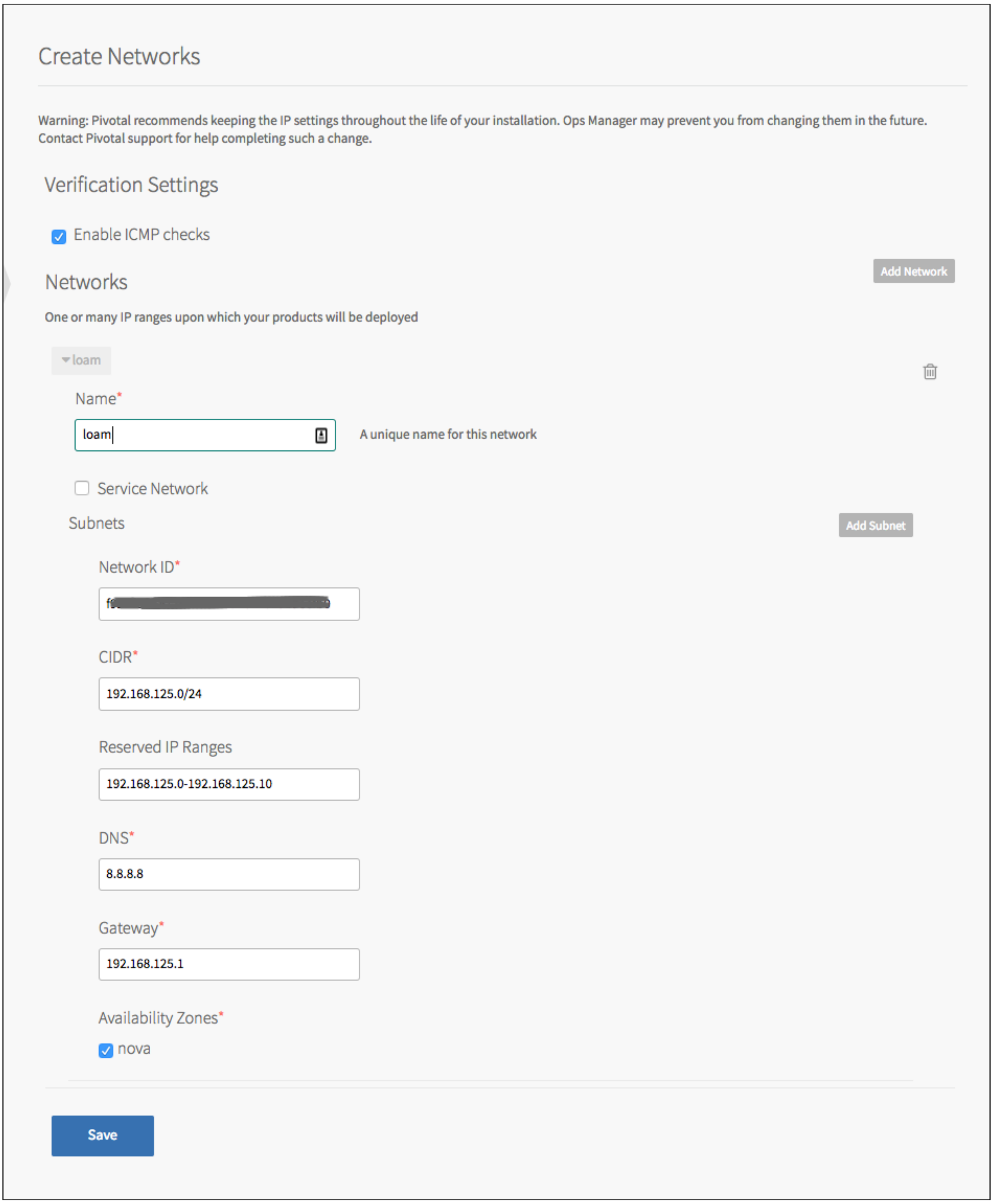
Step 6: Complete the Create Networks Page

1. In the left navigation of your OpenStack dashboard, click **Project > Network > Networks**.

2. Click the name of the network that contains the private subnet where you deployed the Ops Manager VM. The OpenStack Network Detail page displays your network settings.



3. In Ops Manager, select **Create Networks**.



4. Select **Enable ICMP checks** to enable ICMP on your networks. Ops Manager uses ICMP checks to confirm that components within your network are reachable. Review the Configure Security topic to ensure you have setup ICMP in your Security Group.

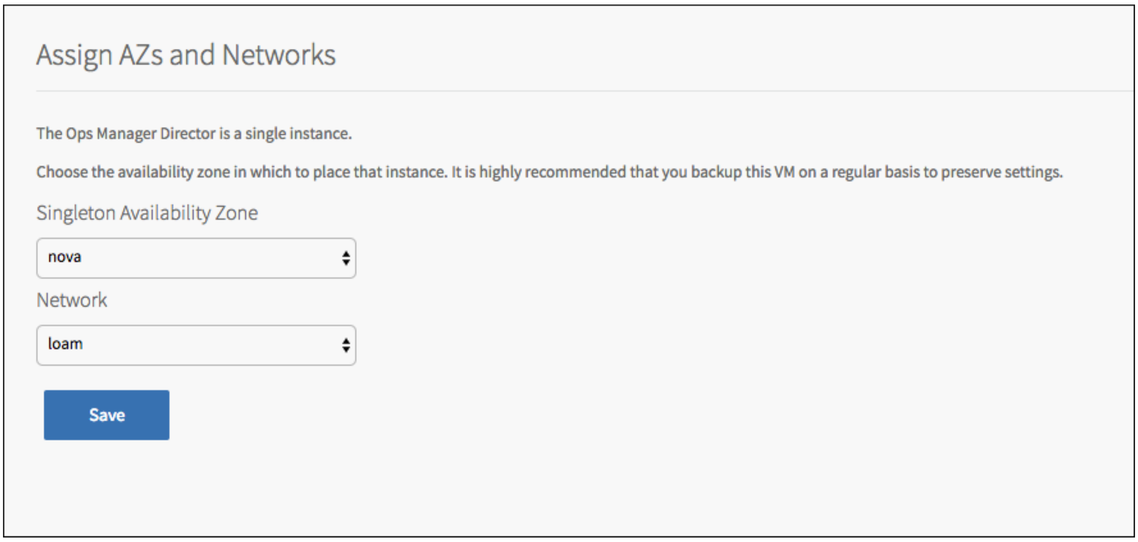
5. Use the following steps to create one or more Ops Manager networks using information from your OpenStack network:
   - Click **Add Network**.
   - Enter a unique **Name** for the network.
   - If you want to dynamically provision VMs in this network for use with on-demand services, select the **Service Networks** checkbox. When the checkbox is selected, Ops Manager does not provision VMs within the specified CIDR range.
   - Click **Add Subnet** to create one or more subnets for the network.
   - For **Network ID**, use the **ID** from the OpenStack page.
   - For **CIDR**, use the **Network Address** from the OpenStack page.
   - For **Reserved IP Ranges**, use the first 10 IP addresses of the **Network Address** range, and the private IP address of the Ops Manager instance that you recorded in the Associate a Floating IP Address step of the Provisioning the OpenStack Infrastructure topic.

- For **DNS**, enter one or more Domain Name Servers.
- For **Gateway**, use the **Gateway IP** from the OpenStack page.
- For **Availability Zones**, select which Availability Zones to use with the network.

6. Click **Save**.

Step 7: Complete the Assign AZs and Networks Page

1. Select **Assign Availability Zones**.

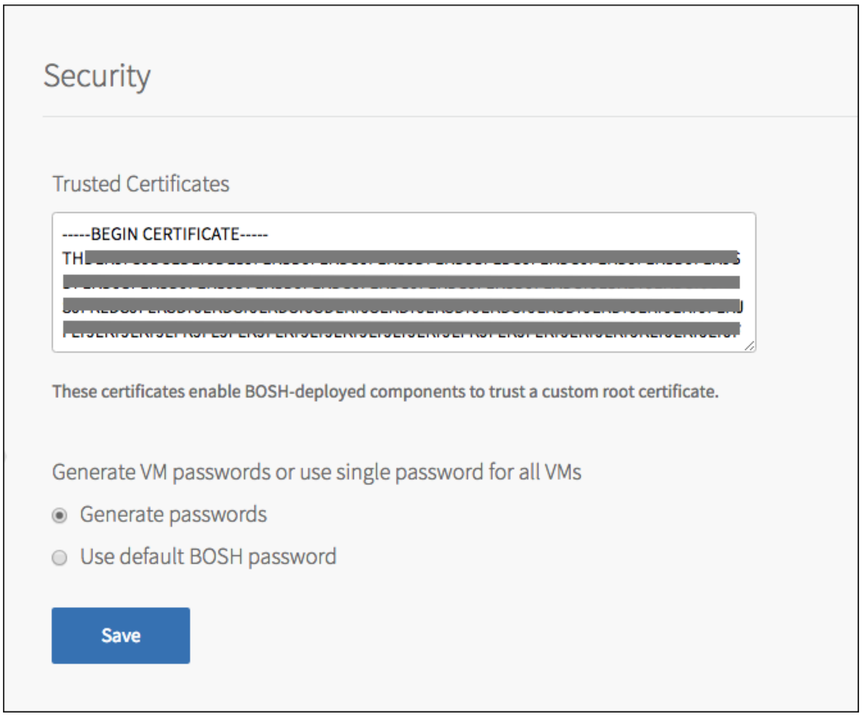


2. From the **Singleton Availability Zone** drop-down menu, select the availability zone that you created in a previous step. The Ops Manager Director installs in this Availability Zone.

3. Use the drop-down menu to select the **Network** that you created in a previous step. Ops Manager Director installs in this network.

4. Click **Save**.

Step 8: Complete the Security Page



1. Select **Security**.

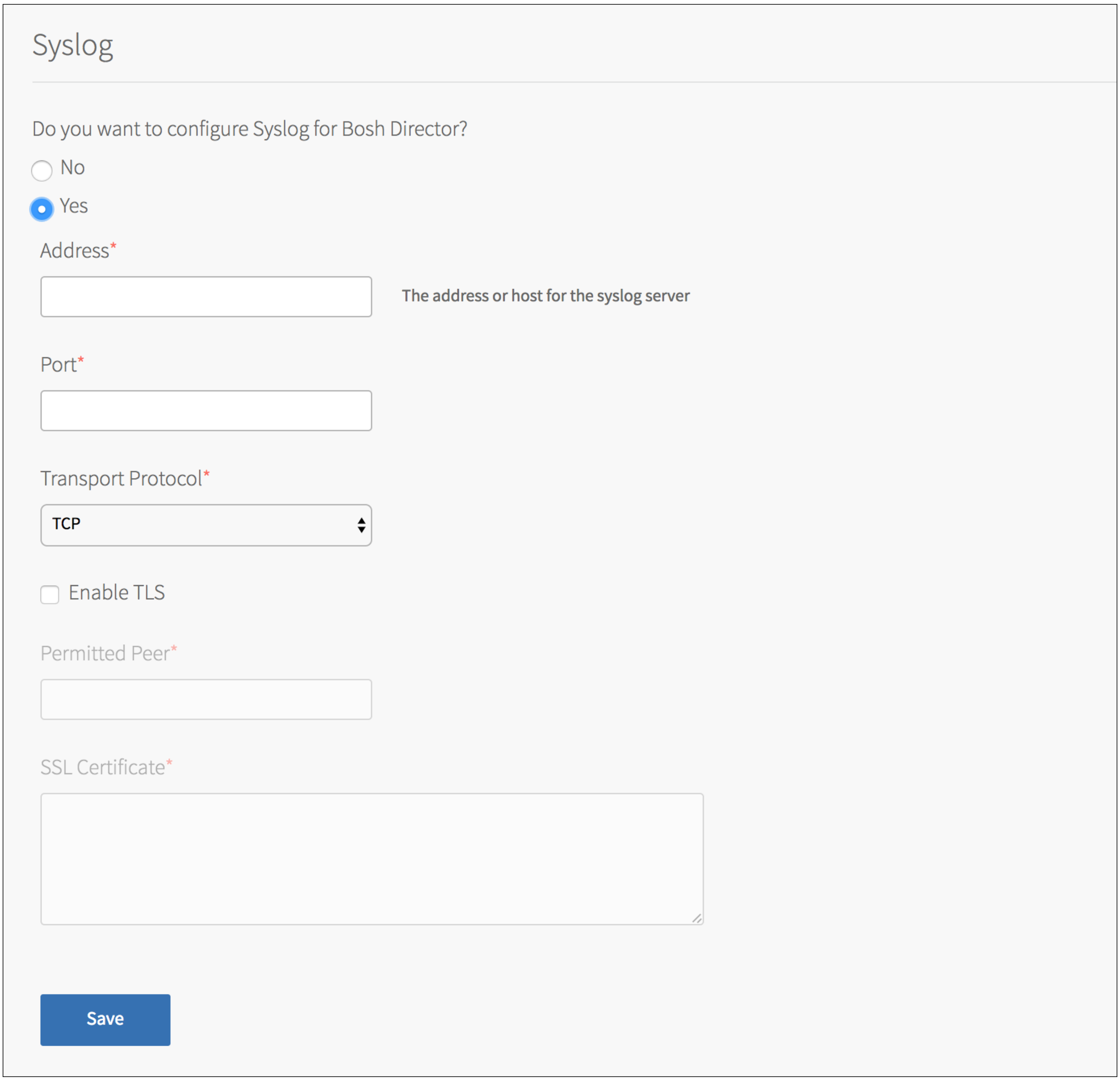
2. In **Trusted Certificates**, enter a custom certificate authority (CA) certificate to insert into your organization's certificate trust chain. This feature enables all BOSH-deployed components in your deployment to trust a custom root certificate. If you want to use Docker Registries for running app instances in Docker containers, use this field to enter your certificate for your private Docker Registry. See the Using Docker Registries topic for more information.

3. Choose **Generate passwords** or **Use default BOSH password**. Pivotal recommends that you use the **Generate passwords** option for greater security.

4. Click **Save**. To view your saved Director password, click the **Credentials** tab.

Step 9: Complete the Syslog Page

1. Select **Syslog**.



2. (Optional) To send BOSH Director system logs to a remote server, select **Yes**.

3. In the **Address** field, enter the IP address or DNS name for the remote server.

4. In the **Port** field, enter the port number that the remote server listens on.

5. In the **Transport Protocol** dropdown menu, select **TCP**, **UDP**, or **RELP**. This selection determines which transport protocol is used to send the logs

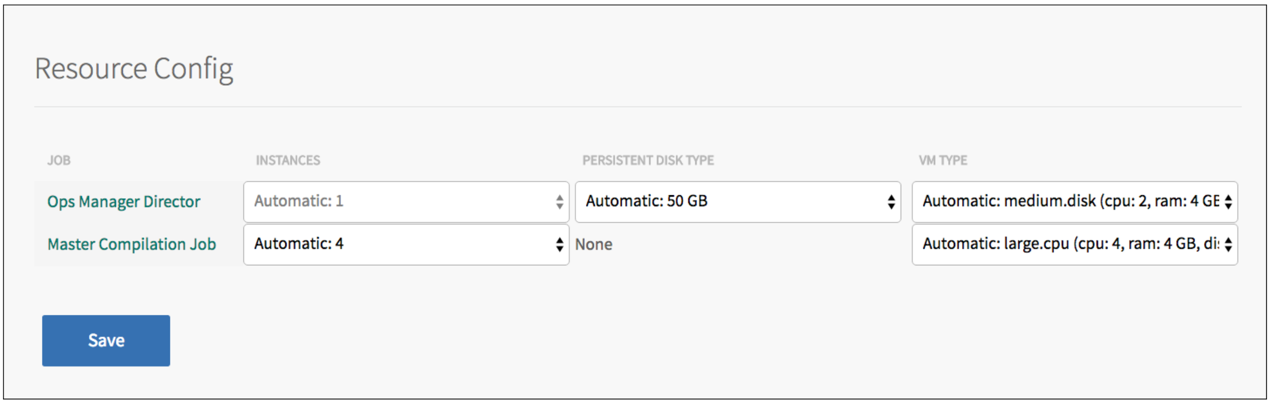
to the remote server.

6. (Optional) Mark the **Enable TLS** checkbox to use TLS encryption when sending logs to the remote server.
   - In the **Permitted Peer** field, enter either the name or SHA1 fingerprint of the remote peer.
   - In the **SSL Certificate** field, enter the SSL certificate for the remote server.

7. Click **Save**.

Step 10: Complete the Resource Config Page

1. Select **Resource Config**.



2. Adjust any values as necessary for your deployment, such as increasing the persistent disk size. Select**Automatic** from the drop-down menu to provision the amount of persistent disk predefined by the job. If the persistent disk field reads **None**, the job does not require persistent disk space.

   > **Note**: Ops Manager v1.11 and above require a Director VM with at least 8 GB memory.

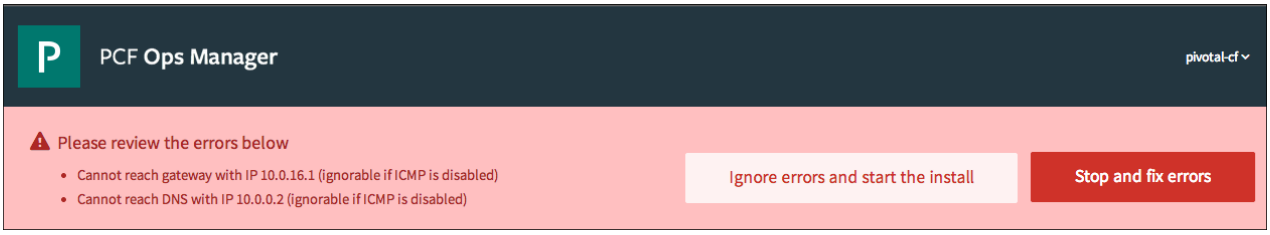
   > **Note**: If you set a field to **Automatic** and the recommended resource allocation changes in a future version, Ops Manager automatically uses the updated recommended allocation.

3. Click **Save**.

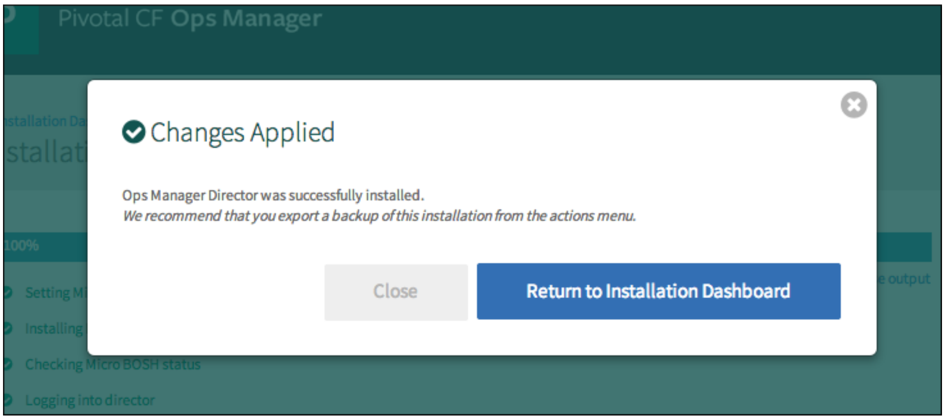
Step 11: Complete Ops Manager Director Installation

1. Click the **Installation Dashboard** link to return to the Installation Dashboard.

2. Click **Apply Changes**. If the following ICMP error message appears, click **Ignore errors and start the install**.



3. Ops Manager Director installs. The image shows the **Changes Applied** message that Ops Manager displays when the installation process successfully completes.



4. After you complete this procedure, follow the instructions in the Installing PAS after Deploying PCF on OpenStack topic.

Return to Installing Pivotal Cloud Foundry on OpenStack.

Installing PAS after Deploying PCF on OpenStack

Page last updated:

This topic describes how to install and configure Pivotal Application Service (PAS) after deploying Pivotal Cloud Foundry (PCF) on OpenStack.

Use this topic when Installing Pivotal Cloud Foundry on OpenStack.

Before beginning this procedure, ensure that you have successfully completed all steps in the Provisioning the OpenStack Infrastructure topic and the Configuring Ops Manager Director after Deploying Pivotal Cloud Foundry on OpenStack topics.

Step 1: Add PAS to Ops Manager

1. If you have not already downloaded PAS, log in to Pivotal Network, and click PAS.

2. From the **Releases** drop-down, select the release to install and choose one of the following:

   a. Click PAS to download the PAS `.pivotal` file
   b. Click **PCF Small Footprint Runtime** to download the Small Footprint Runtime `.pivotal` file. For more information, see Getting Started with Small Footprint Runtime.

3. Navigate to the Pivotal Cloud Foundry Operations Manager Installation Dashboard.

4. Click the Pivotal Network link on the left to add PAS to Ops Manager. For more information, refer to the Adding and Deleting Products topic.

Step 2: Assign Availability Zones and Networks

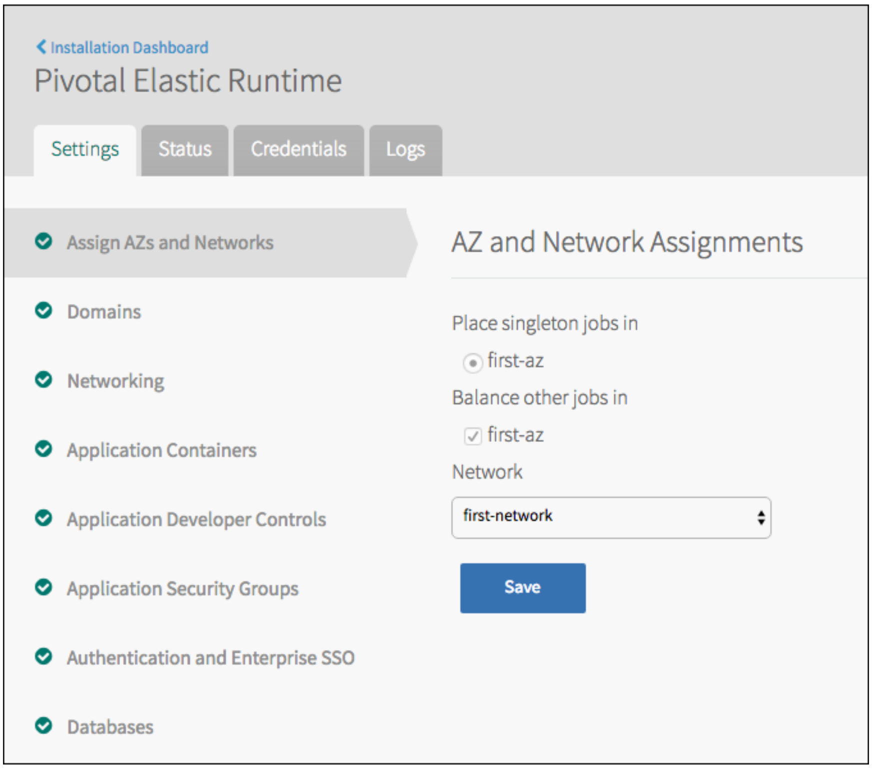
Note: Pivotal recommends at least three Availability Zones for a highly available installation of PAS.

1. Select **Assign AZ and Networks**. These are the Availability Zones that you create when configuring Ops Manager Director.

2. Select an Availability Zone under **Place singleton jobs**. Ops Manager runs any job with a single instance in this Availability Zone.

3. Select one or more Availability Zones under **Balance other jobs**. Ops Manager balances instances of jobs with more than one instance across the Availability Zones that you specify.

4. From the **Network** drop-down box, choose the network on which you want to run PAS.

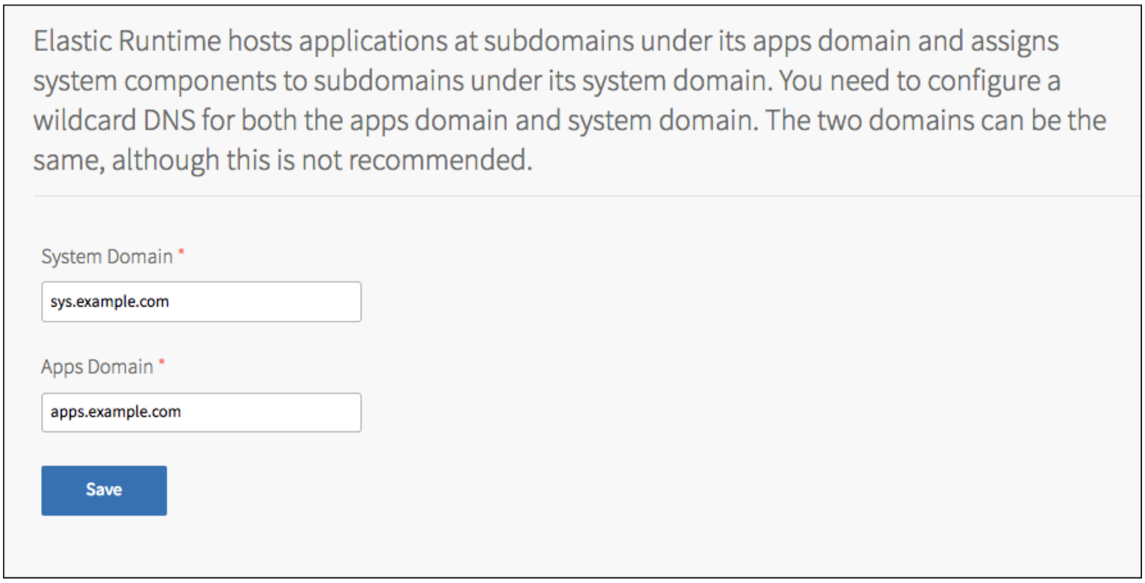


5. Click **Save**.

Note: When you save this form, a verification error displays because the PCF security group blocks ICMP. You can ignore this error.

Step 3: Configure Domains

1. Select **Domains**.



2. Enter the system and application domains.
   - The **System Domain** defines your target when you push apps to PAS.
   - The **Apps Domain** defines where PAS should serve your apps.

> **Note**: Pivotal recommends that you use the same domain name but different subdomain names for your system and app domains. Doing so allows you to use a single wildcard certificate for the domain while preventing apps from creating routes that overlap with system routes. For example, name your system domain `system.EXAMPLE.com` and your apps domain `apps.EXAMPLE.com`.

> **Note**: You configured wildcard DNS records for these domains in an earlier step.

3. Click **Save**.

Step 4: Configure Networking

1. Select **Networking**.

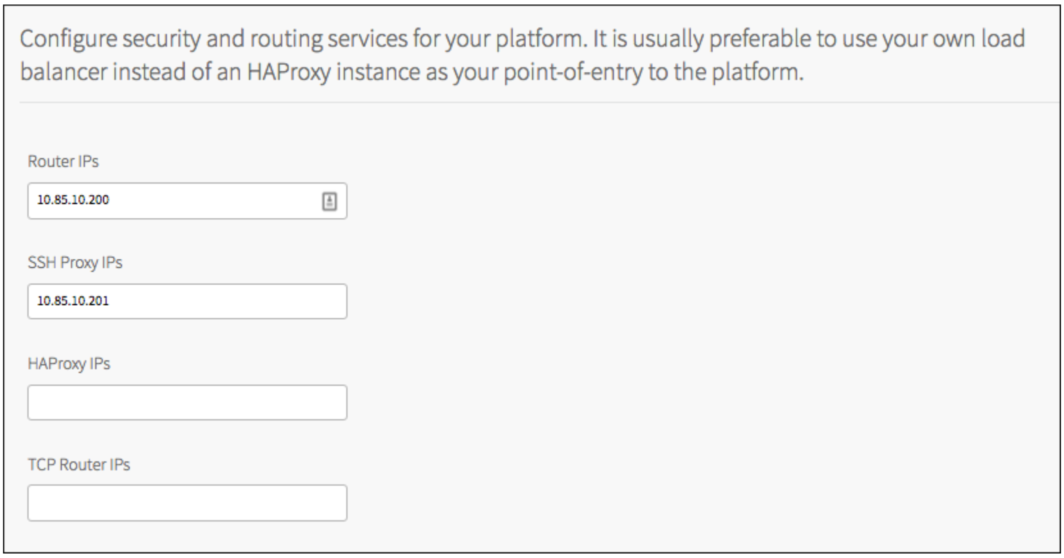
2. The values you enter in the **Router IPs** and **HAProxy IPs** fields depend on whether you are using HAProxy in your deployment. Use the table below to determine how to complete these fields.

> **Note**: If you choose to assign specific IP addresses in either the **Router IPs** or **HAProxy IPs** field, ensure that these IP addresses are in the subnet that you configured for PAS in Ops Manager.

Using HAProxy?	Router IPs Field	HAProxy IPs Field
No	1. Choose IP addresses from the subnet you configured in Ops Manager. 2. Enter these IP addresses in the **Router IPs** field. You should specify more than one IP address for high availability. 3. Configure your load balancer to forward requests for the domains that you have configured for your deployment to these IP addresses.	Leave this field blank.
Yes	Leave this field blank.	1. Choose IP addresses from the subnet you configured in Ops Manager. 2. Enter these IP addresses in the **HAProxy IPs** field. You should specify more than one IP address for high availability. 3. Configure your load balancer to forward requests for the domains you have configured for your deployment to these IP addresses.

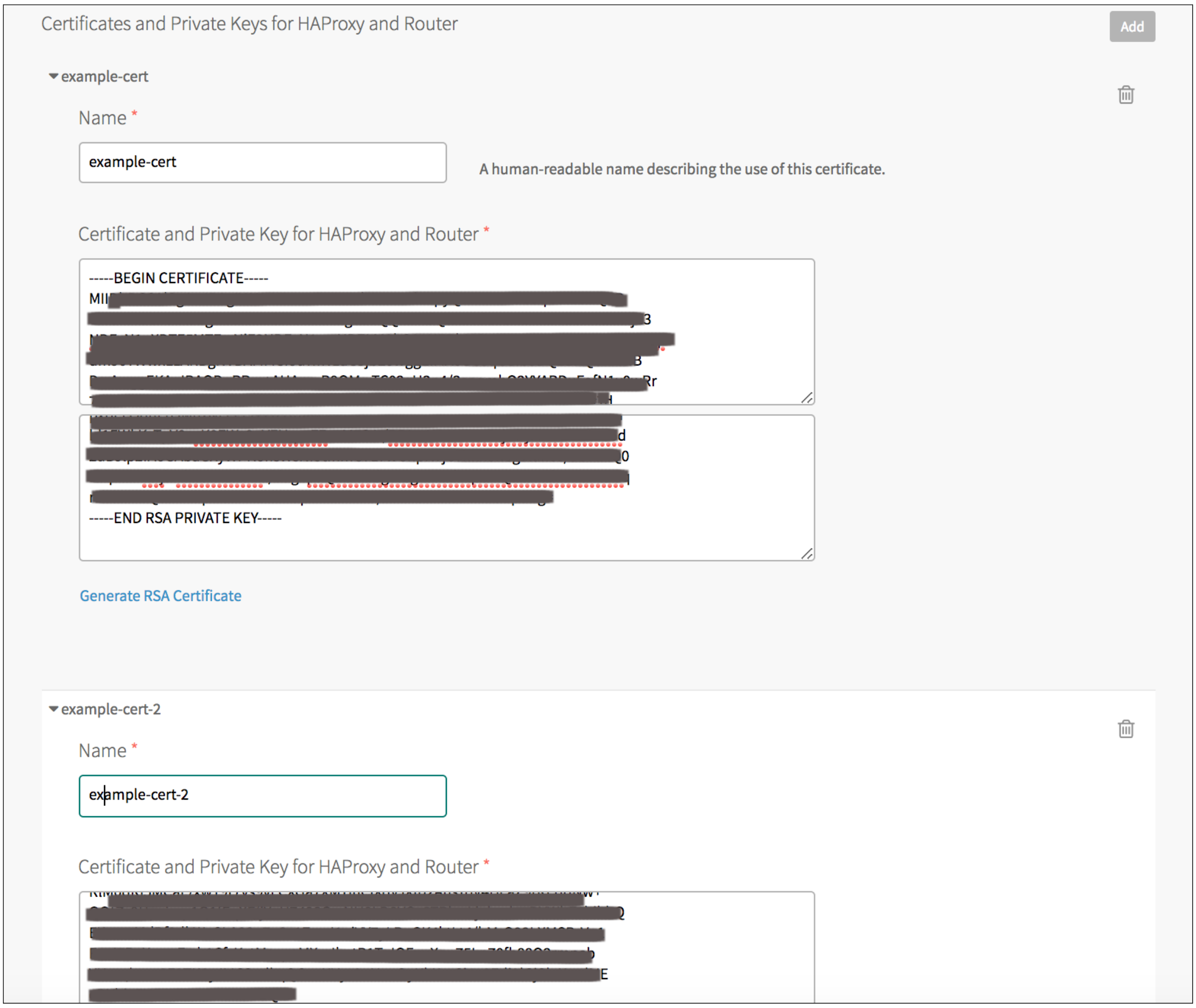
3. (Optional) In **SSH Proxy IPs**, add the IP address for your Diego Brain, which will accept requests to SSH into application containers on port `2222`.

4. (Optional) In **TCP Router IPs**, add the IP address(es) you would like assigned to the TCP Routers. You enable this feature at the bottom of this screen.



5. Under **Certificates and Private Key for HAProxy and Router**, you must provide at least one **Certificate and Private Key** name and certificate keypair for HAProxy and Gorouter. The HAProxy and Gorouter are enabled to receive TLS communication by default. You can configure multiple certificates for HAProxy and Gorouter.

    a. Click the **Add** button to add a name for the certificate chain and its private keypair. This certificate is the default used by Gorouter and HAProxy.

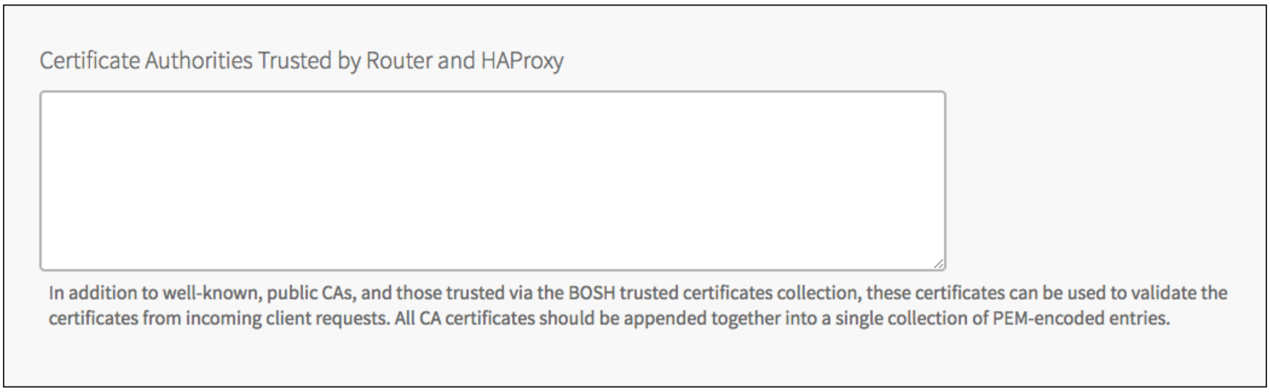
    

    You can either provide a certificate signed by a Certificate Authority (CA) or click on the **Generate RSA Certificate** link to generate a self-signed certificate in Ops Manager.

    b. If you want to configure multiple certificates for HAProxy and Gorouter, click the **Add** button and fill in the appropriate fields for each additional certificate keypair.

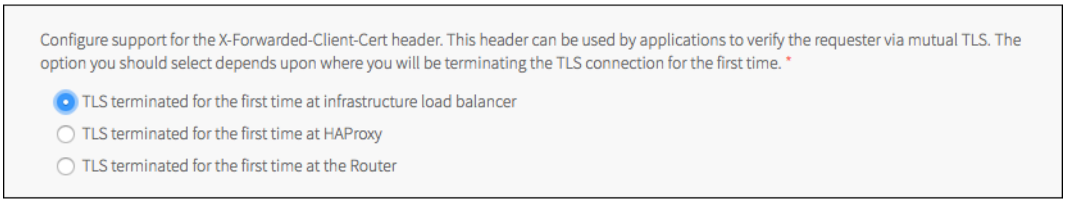
For details about generating certificates in Ops Manager for your wildcard system domains, see the [Providing a Certificate for Your SSL/TLS Termination Point]() topic.

6. (Optional) When validating client requests using mutual TLS, the Gorouter trusts multiple certificate authorities (CAs) by default. If you want to configure the Gorouter and HAProxy to trust additional CAs, enter your CA certificates under **Certificate Authorities Trusted by Router and HAProxy**. All CA certificates should be appended together into a single collection of PEM-encoded entries.



7. In the **Minimum version of TLS supported by HAProxy and Router** field, select the minimum version of TLS to use in HAProxy and Router communications. HAProxy and Router use TLS v1.2 by default. If you need to accommodate clients that use an older version of TLS, select a lower minimum version. For a list of TLS ciphers supported by the Gorouter, see [Securing Traffic into Cloud Foundry]().

8. Under **Configure support for the X-Forwarded-Client-Cert header**, configure PCF handles `x-forwarded-client-cert` (XFCC) HTTP headers based on where TLS is terminated for the first time in your deployment.



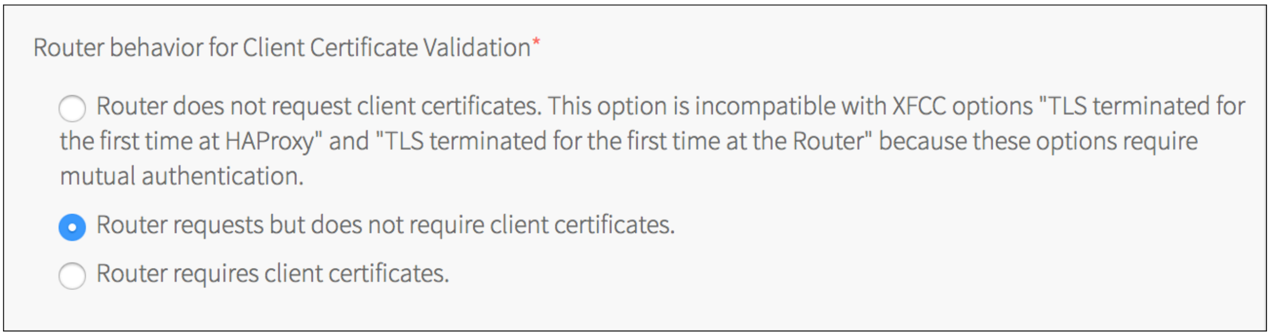
The following table indicates which option to choose based on your deployment layout.

If your deployment is configured as follows:	Then select the following option:	Additional notes:
◦ The Load Balancer is terminating TLS, and ◦ Load balancer is configured to put the client certificate from a mutual authentication TLS handshake into the X-Forwarded-Client-Cert HTTP header	TLS terminated for the first time at infrastructure load balancer (default).	Both HAProxy and the Gorouter forward the XFCC header when included in the request.
◦ The Load Balancer is configured to pass through the TLS handshake via TCP to the instances of HAProxy, and ◦ HAProxy instance count is > 0	TLS terminated for the first time at HAProxy.	HAProxy sets the XFCC header with the client certificate received in the TLS handshake. The Gorouter forwards the header. **Breaking Change**: In the **Router behavior for Client Certificates** field, you cannot select the **Router does not request client certificates** option.
		The Gorouter strips the XFCC header if it is included in the request and forwards the client certificate received in the TLS handshake in a new

◦ The Load Balancer is configured to pass through the TLS handshake via TCP to instances of the Gorouter	TLS terminated for the first time at the Gorouter.	XFCC header. If you have deployed instances of HAProxy, app traffic bypasses those instances in this configuration. If you have also configured your load balancer to route requests for ssh directly to the Diego Brain, consider reducing HAProxy instances to 0. **Breaking Change**: In the **Router behavior for Client Certificates** field, you cannot select the **Router does not request client certificates** option.

For a description of the behavior of each configuration option, see [Forward Client Certificate to Applications](#).

9. To configure Gorouter behavior for handling client certificates, select one of the following options in the **Router behavior for Client Certificate Validation** field.



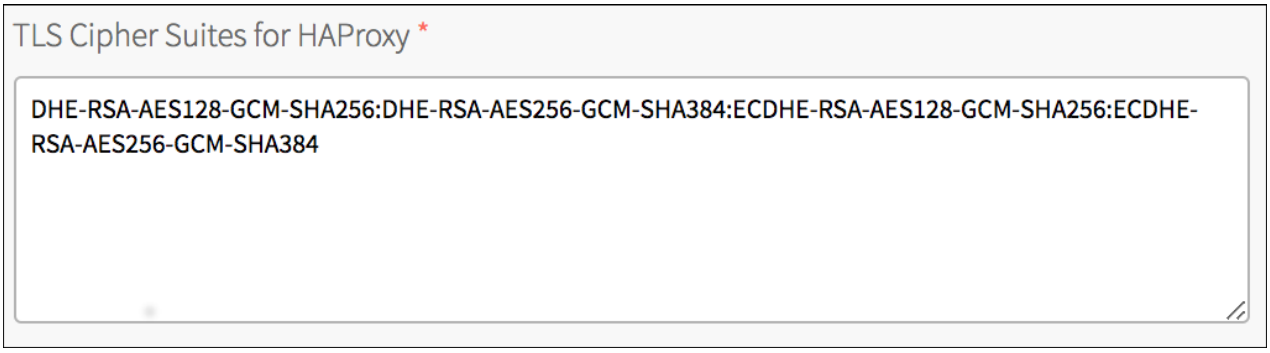
   - **Router does not request client certificates.** This option is incompatible with the XFCC configuration options **TLS terminated for the first time at HAProxy** and **TLS terminated for the first time at the Router** in PAS because these options require mutual authentication. As client certificates are not requested, client will not provide them, and thus validation of client certificates will not occur.
   - **Router requests but does not require client certificates.** Gorouter requests client certificates in TLS handshakes, validates them when presented, but does not require them. This is the default configuration.
   - **Router requires client certificates.** Gorouter validates that the client certificate is signed by a Certificate Authority that Gorouter trusts. If Gorouter cannot validate the client certificate, the TLS handshake fails.

10. In the **TLS Cipher Suites for Router** field, specify the TLS cipher suites to use for TLS handshakes between the Gorouter and downstream clients like load balancers or HAProxy. Use an ordered, colon-delimited list of Golang-supported TLS cipher suites in the OpenSSL format. The recommended setting is `ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384`. Operators should verify that the ciphers are supported by any clients or downstream components that will initiate TLS handshakes with the Gorouter. For a list of TLS ciphers supported by the Gorouter, see [Securing Traffic into Cloud Foundry](#).

Verify that whatever client is participating in the TLS handshake with the Gorouter has at least one cipher suite in common with the Gorouter.

> **Note**: Specify cipher suites that are supported by the versions configured in the **Minimum version of TLS supported by HAProxy and Router** field.

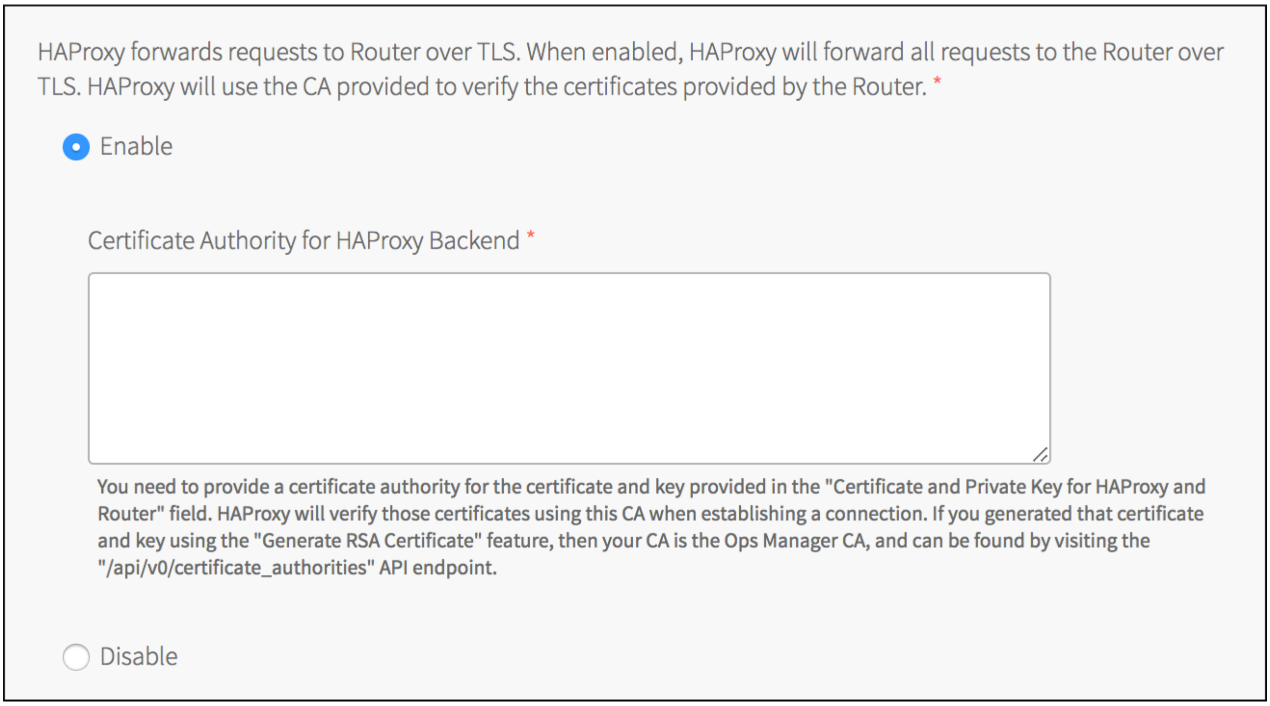
11. In the **TLS Cipher Suites for HAProxy** field, specify the TLS cipher suites to use in HAProxy for TLS handshakes between HAProxy and its clients such as load balancers and Gorouter. Use an ordered, colon-delimited list of TLS cipher suites in the OpenSSL format. The recommended setting: `DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384`
Operators should verify that the ciphers are supported by any clients or downstream components that will initiate TLS handshakes with the HAProxy.



Verify that whatever clients are participating in the TLS handshake with HAProxy have at least one cipher suite in common with HAProxy.

> **Note:** Specify cipher suites that are supported by the versions configured in the **Minimum version of TLS supported by HAProxy and Router** field.

12. Under **HAProxy forwards requests to Router over TLS**, select **Enable** or **Disable** based on your deployment layout.



- **Enable HAProxy forwarding of requests to Router over TLS**

If you want to:	Encrypt communication between HAProxy and the Gorouter
Then configure the following:	1. Leave **Enable** selected. 2. In the **Certificate Authority for HAProxy Backend** field, specify the Certificate Authority (CA) that signed the certificate you configured in the **Certificate and Private Key for HAProxy and Router** field. **Note:** If you used the **Generate RSA Certificate** link to generate a self-signed certificate, then the CA to specify is the Ops Manager CA, which you can locate at the CA endpoint in the Ops Manager API. 3. Make sure that Gorouter and HAProxy have TLS cipher suites in common in the **TLS Cipher Suites for Router** and **TLS Cipher Suites for HAProxy** fields.
See also:	◦ Terminating SSL/TLS at the Load Balancer and Gorouter ◦ Providing a Certificate for Your SSL/TLS Termination Point ◦ Using the Ops Manager API

- **Disable HAProxy forwarding of requests to Router over TLS**

If you want to:	Use non-encrypted communication between HAProxy and Gorouter, or you are not using HAProxy

Then configure the following:	1. Select **Disable**. 2. If you are not using HAProxy, set the number of HAProxy job instances to `0` on the **Resource Config** page. See Disable Unused Resources.
See also:	◦ Terminating SSL/TLS at the Gorouter Only ◦ Terminating SSL/TLS at the Load Balancer Only

13. If you are not using SSL encryption or if you are using self-signed certificates, select **Disable SSL certificate verification for this environment**. Selecting this checkbox also disables SSL verification for route services.

    > **Note**: For production deployments, Pivotal does not recommend disabling SSL certificate verification.

14. (Optional) If you want HAProxy or the Gorouter to reject any HTTP (non-encrypted) traffic, select the **Disable HTTP on HAProxy and Gorouter** checkbox. When selected, HAProxy and Gorouter will not listen on port 80.

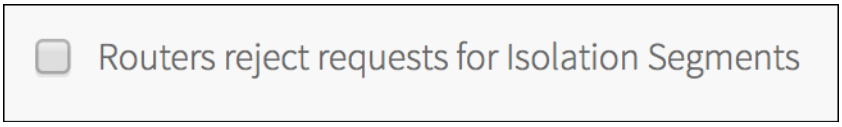
    

15. (Optional) Select the **Disable insecure cookies on the Router** checkbox to set the secure flag for cookies generated by the router.

16. (Optional) To disable the addition of Zipkin tracing headers on the Gorouter, deselect the **Enable Zipkin tracing headers on the router** checkbox. Zipkin tracing headers are enabled by default. For more information about using Zipkin trace logging headers, see Zipkin Tracing in HTTP Headers.

17. (Optional) To stop the Router from writing access logs to local disk, deselect the **Enable Router to write access logs locally** checkbox. You should consider disabling this checkbox for high traffic deployments since logs may not be rotated fast enough and can fill up the disk.

18. By default, the PAS routers handle traffic for applications deployed to an isolation segment created by the PCF Isolation Segment tile. To configure the PAS routers to reject requests for applications within isolation segments, select the **Routers reject requests for Isolation Segments** checkbox.

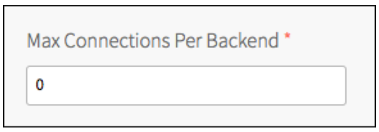
    

    Do not enable this option without deploying routers for each isolation segment. See the following topics for more information:
    - Installing PCF Isolation Segment
    - Sharding Routers for Isolation Segments

19. In the **Choose whether or not to enable route services** section, choose either **Enable route services** or **Disable route services**. Route services are a class of marketplace services that perform filtering or content transformation on application requests and responses. See the Route Services topic for details.

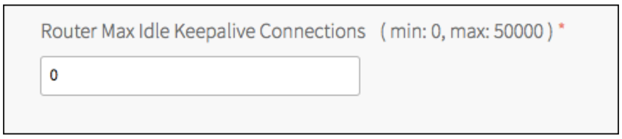
20. (Optional) If you want to limit the number of app connections to the backend, enter a value in the **Max Connections Per Backend** field. You can use this field to prevent a poorly behaving app from all the connections and impacting other apps.

    To choose a value for this field, review the peak concurrent connections received by instances of the most popular apps in your deployment. You can determine the number of concurrent connections for an app from the `httpStartStop` event metrics emitted for each app request.

    If your deployment uses PCF Metrics, you can also obtain this peak concurrent connection information from Network Metrics. The default value is
    

    `500`.

21. Enter a value for **Router Max Idle Keepalive Connections**. See Considerations for Configuring max_idle_connections.

22. (Optional) To accommodate larger uploads over connections with high latency, increase the number of seconds in the **Router Timeout to Backends** field.

23. (Optional) Use the **Frontend Idle Timeout for Gorouter and HAProxy** field to help prevent connections from your load balancer to Gorouter or

HAProxy from being closed prematurely. The value you enter sets the duration, in seconds, that Gorouter or HAProxy maintains an idle open connection from a load balancer that supports keep-alive.

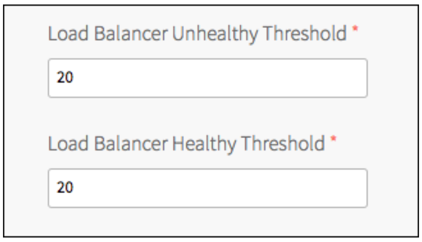
In general, set the value higher than your load balancer's backend idle timeout to avoid the race condition where the load balancer sends a request before it discovers that Gorouter or HAProxy has closed the connection.

See the following table for specific guidance and exceptions to this rule:

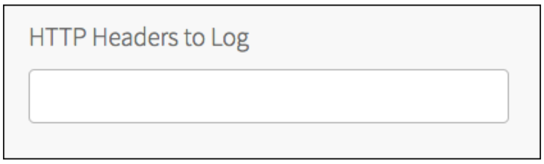
IaaS	Guidance
AWS	AWS ELB has a default timeout of 60 seconds, so Pivotal recommends a value greater than `60`.
Azure	By default, Azure load balancer times out at 240 seconds without sending a TCP RST to clients, so as an exception, Pivotal recommends a value lower than `240` to force the load balancer to send the TCP RST.
GCP	GCP has a default timeout of 600 seconds, so Pivotal recommends a value greater than `600`.
Other	Set the timeout value to be greater than that of the load balancer's backend idle timeout.

24. (Optional) Increase the value of **Load Balancer Unhealthy Threshold** to specify the amount of time, in seconds, that the router continues to accept connections before shutting down. During this period, healthchecks may report the router as unhealthy, which causes load balancers to failover to other routers. Set this value to an amount greater than or equal to the maximum time it takes your load balancer to consider a router instance unhealthy, given contiguous failed healthchecks.

25. (Optional) Modify the value of **Load Balancer Healthy Threshold**. This field specifies the amount of time, in seconds, to wait until declaring the Router instance started. This allows an external load balancer time to register the Router instance as healthy.

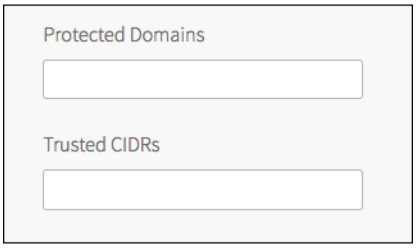


26. (Optional) If app developers in your organization want certain HTTP headers to appear in their app logs with information from the Gorouter, specify them in the **HTTP Headers to Log** field. For example, to support app developers that deploy Spring apps to PCF, you can enter Spring-specific HTTP headers.

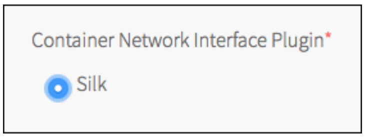


27. If you expect requests larger than the default maximum of 16 Kbytes, enter a new value (in bytes) for **HAProxy Request Max Buffer Size**. You may need to do this, for example, to support apps that embed a large cookie or query string values in headers.

28. If your PCF deployment uses HAProxy and you want it to receive traffic only from specific sources, use the following fields:
    - **Protected Domains:** Enter a comma-separated list of domains from which PCF can receive traffic.
    - **Trusted CIDRs:** Optionally, enter a space-separated list of CIDRs to limit which IP addresses from the **Protected Domains** can send traffic to PCF.



29. The **Loggregator Port** defaults to `443` if left blank. Enter a new value to override the default.



30. For **Container Network Plugin Interface**, ensure **Silk** is selected and review the following fields:

> **Note**: The **External** option exists to support NSX-T integration for vSphere deployments.

a. (Optional) You can change the value in the **Applications Network Maximum Transmission Unit (MTU)** field. Pivotal recommends setting the MTU value for your application network to `1454`. Some configurations, such as networks that use GRE tunnels, may require a smaller MTU value.
b. (Optional) Enter an IP range for the overlay network in the **Overlay Subnet** box. If you do not set a custom range, Ops Manager uses `10.255.0.0/16`.

> **WARNING**: The overlay network IP range must not conflict with any other IP addresses in your network.

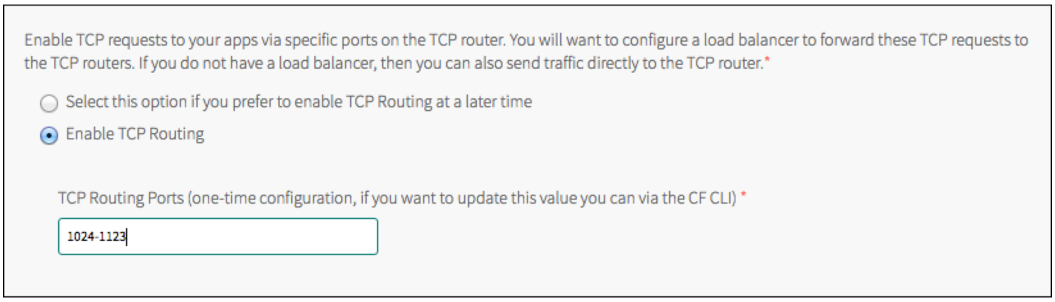
c. Enter a UDP port number in the **VXLAN Tunnel Endpoint Port** box. If you do not set a custom port, Ops Manager uses 4789.
d. For **Denied logging interval**, set the per-second rate limit for packets blocked by either a container-specific networking policy or by Application Security Group rules applied across the space, org, or deployment. This field defaults to `1`.
e. For **UDP logging interval**, set the per-second rate limit for UDP packets sent and received. This field defaults to `100`.
f. To enable logging for app traffic, select **Log traffic for all accepted/denied application packets**. See Manage Logging for Container-to-Container Networking for more information.

31. (Optional) TCP Routing is disabled by default. To enable this feature, perform the following steps:

a. Select **Enable TCP Routing**.
b. In **TCP Routing Ports**, enter a range of ports to be allocated for TCP Routes.

For each TCP route you want to support, you must reserve a range of ports. This is the same range of ports you configured your load balancer with in the Pre-Deployment Steps, unless you configured DNS to resolve the TCP domain name to the TCP router directly.

The **TCP Routing Ports** field accepts a comma-delimited list of individual ports and ranges, for example `1024-1099,30000,60000-60099`. Configuration of this field is only applied on the first deploy, and update updates to the port range are made using the cf CLI. For details about modifying the port range, see the Router Groups topic.

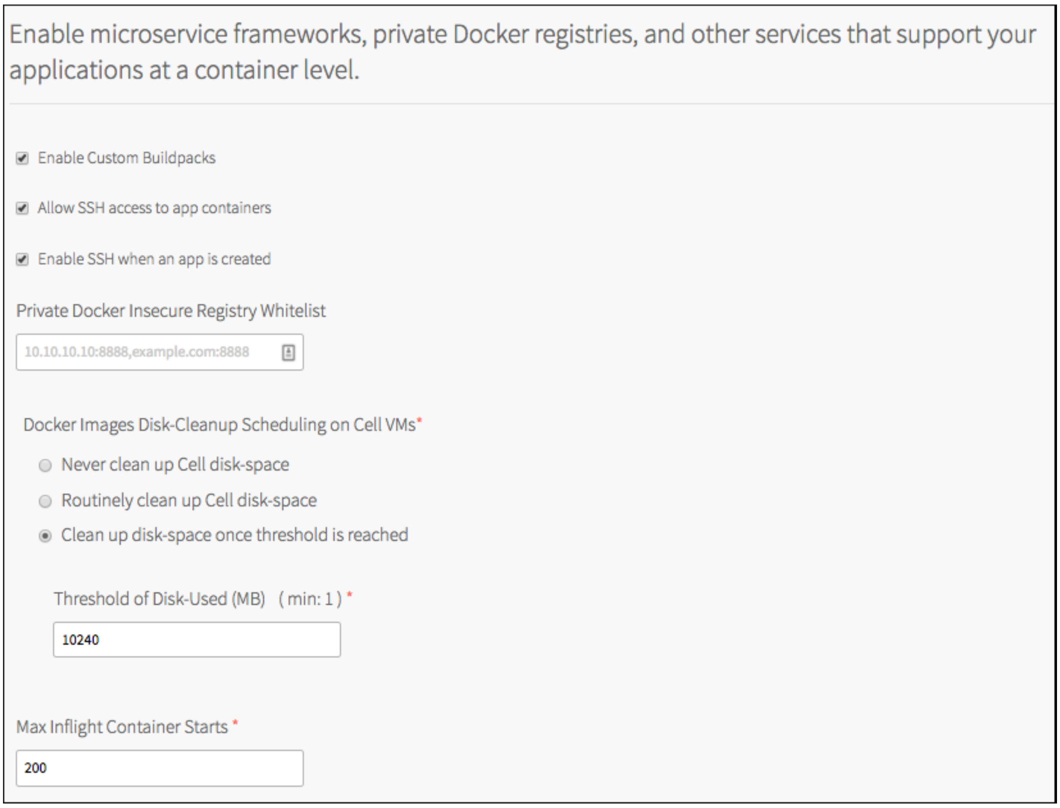


c. Return to the top of the **Networking** screen. In **TCP Router IPs** field, make sure you have entered IP addresses within your subnet CIDR block. These will be the same IP addresses you configured your load balancer with in Pre-Deployment Steps, unless you configured DNS to resolve the TCP domain name directly to an IP you've chosen for the TCP router. You can enter multiple values as a comma-delimited list or as a range. For example, `10.254.0.1, 10.254.0.2` or `10.254.0.1-10.254.0.2`.
d. (Optional) To disable TCP routing, click **Select this option if you prefer to enable TCP Routing at a later time** For more information, see the Configuring TCP Routing in PAS topic.

32. Click **Save**.

Step 5: Configure Application Containers

1. Select **Application Containers**.



2. The **Enable Custom Buildpacks** checkbox governs the ability to pass a custom buildpack URL to the `-b` option of the `cf push` command. By default, this ability is enabled, letting developers use custom buildpacks when deploying apps. Disable this option by disabling the checkbox. For more information about custom buildpacks, refer to the buildpacks section of the PCF documentation.

3. The **Allow SSH access to app containers** checkbox controls SSH access to application instances. Enable the checkbox to permit SSH access across your deployment, and disable it to prevent all SSH access. See the Application SSH Overview topic for information about SSH access permissions at the space and app scope.

4. If you want enable SSH access for new apps by default in spaces that allow SSH, select **Enable SSH when an app is created**. If you deselect the checkbox, developers can still enable SSH after pushing their apps by running `cf enable-ssh APP-NAME`.

5. You can configure Pivotal Application Service (PAS) to run app instances in Docker containers by supplying their IP address range(s) in the **Private Docker Insecure Registry Whitelist** textbox. See the Using Docker Registries topic for more information.

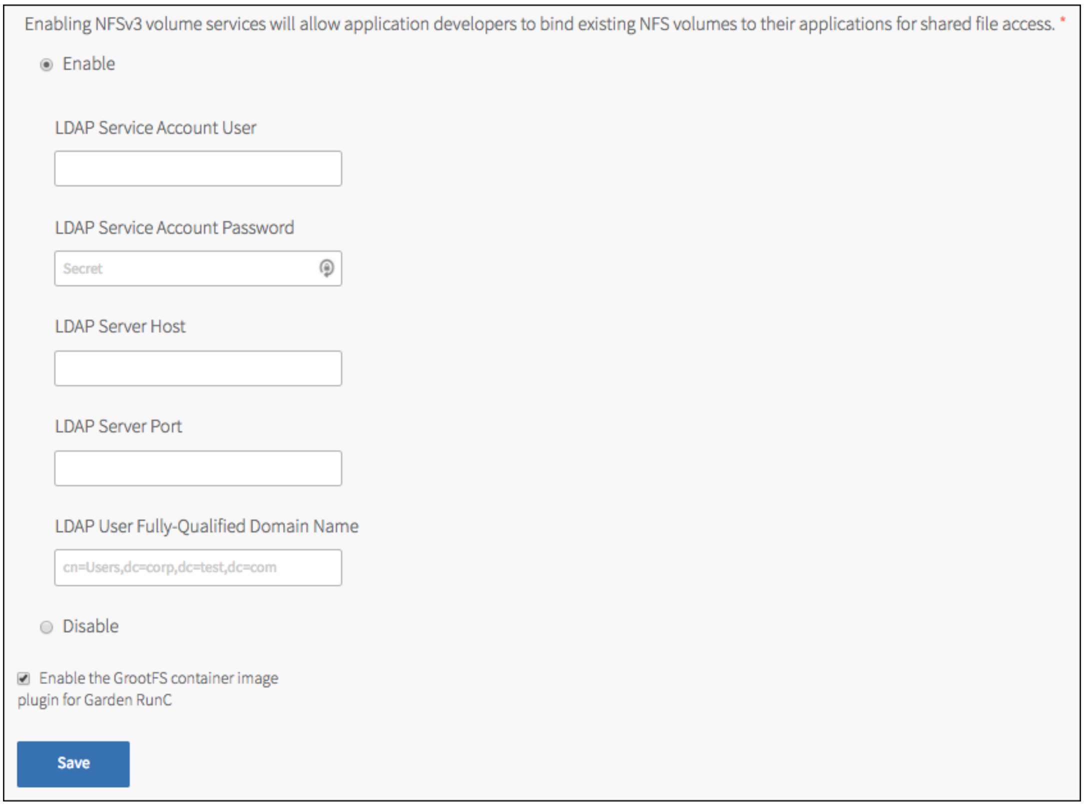
6. Select your preference for **Docker Images Disk-Cleanup Scheduling on Cell VMs**. If you choose **Clean up disk-space once threshold is reached**, enter a **Threshold of Disk-Used** in megabytes. For more information about the configuration options and how to configure a threshold, see Configuring Docker Images Disk-Cleanup Scheduling.

7. Enter a number in the **Max Inflight Container Starts** textbox. This number configures the maximum number of started instances across your deployment's Diego Cells. For more information about this feature, see Setting a Maximum Number of Started Containers.

8. Under **Enabling NFSv3 volume services**, select **Enable** or **Disable**. NFS volume services allow application developers to bind existing NFS volumes to their applications for shared file access. For more information, see the Enabling NFS Volume Services topic.

   > **Note**: In a clean install, NFSv3 volume services is enabled by default. In an upgrade, NFSv3 volume services is set to the same setting as it was in the previous deployment.

9. (Optional) To configure LDAP for NFSv3 volume services, perform the following steps:



- For **LDAP Service Account User**, enter the username of the service account in LDAP that will manage volume services.
- For **LDAP Service Account Password**, enter the password for the service account.
- For **LDAP Server Host**, enter the hostname or IP address of the LDAP server.
- For **LDAP Server Port**, enter the LDAP server port number. If you do not specify a port number, Ops Manager uses 389.
- For **LDAP Server Protocol**, enter the server protocol. If you do not specify a protocol, Ops Manager uses TCP.
- For **LDAP User Fully-Qualified Domain Name**, enter the fully qualified path to the LDAP service account. For example, if you have a service account named `volume-services` that belongs to organizational units (OU) named `service-accounts` and `my-company`, and your domain is named `domain`, the fully qualified path looks like the following:

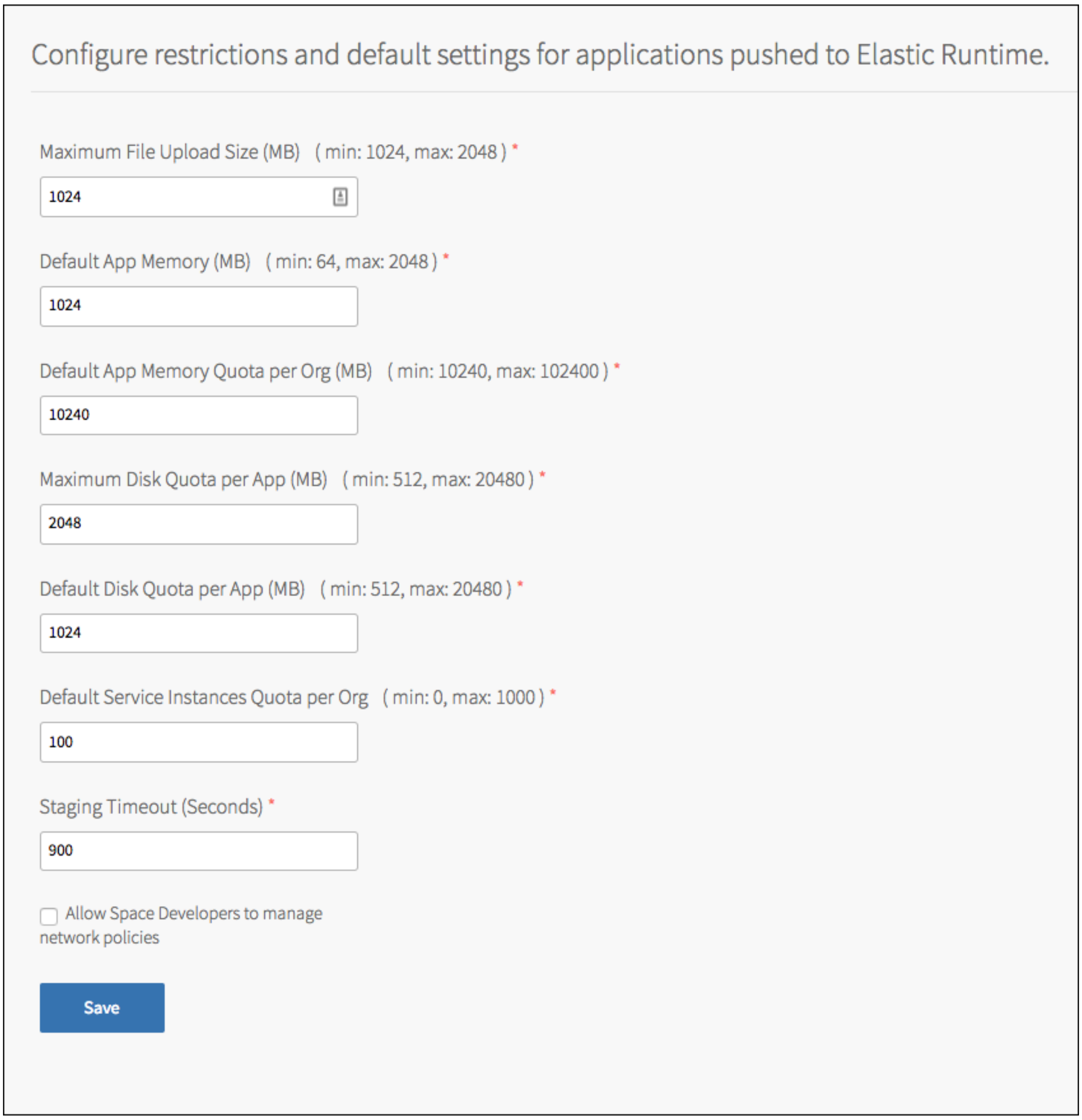
  ```
  CN=volume-services,OU=service-accounts,OU=my-company,DC=domain,DC=com
  ```

10. By default, PAS manages container images using the GrootFS plugin for Garden-runC. If you experience issues with GrootFS, you can disable the plugin and use the image plugin built into Garden-runC.

11. Click **Save**.

Step 6: Configure Application Developer Controls

1. Select **Application Developer Controls**.



2. Enter the **Maximum File Upload Size (MB)**. This is the maximum size of an application upload.

3. Enter the **Default App Memory (MB)**. This is the amount of RAM allocated by default to a newly pushed application if no value is specified with the cf CLI.

4. Enter the **Default App Memory Quota per Org**. This is the default memory limit for all applications in an org. The specified limit only applies to the first installation of PAS. After the initial installation, operators can use the cf CLI to change the default value.

5. Enter the **Maximum Disk Quota per App (MB)**. This is the maximum amount of disk allowed per application.

   > **Note**: If you allow developers to push large applications, PAS may have trouble placing them on Cells. Additionally, in the event of a system upgrade or an outage that causes a rolling deploy, larger applications may not successfully re-deploy if there is insufficient disk capacity. Monitor your deployment to ensure your Cells have sufficient disk to run your applications.

6. Enter the **Default Disk Quota per App (MB)**. This is the amount of disk allocated by default to a newly pushed application if no value is specified with the cf CLI.

7. Enter the **Default Service Instances Quota per Org**. The specified limit only applies to the first installation of PAS. After the initial installation, operators can use the cf CLI to change the default value .

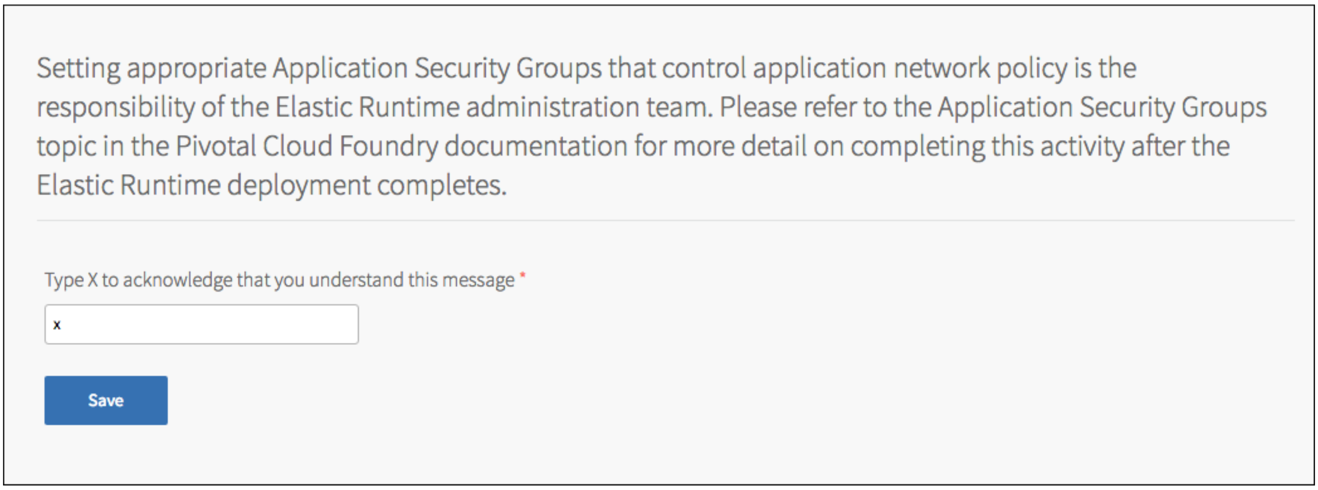
8. Enter the **Staging Timeout (Seconds)**. When you stage an application droplet with the Cloud Controller, the server times out after the number of seconds you specify in this field.

9. Select the **Allow Space Developers to manage network policies** checkbox to permit developers to manage their own network policies for their applications.

10. Click **Save**.

Step 7: Review Application Security Groups

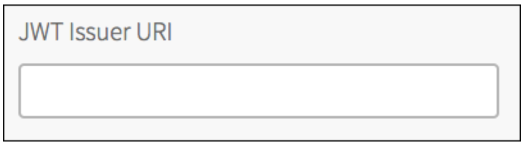
Setting appropriate Application Security Groups is critical for a secure deployment. Type `X` in the box to acknowledge that once the Pivotal Application Service (PAS) deployment completes, you will review and set the appropriate application security groups. See Restricting App Access to Internal PCF Components for instructions.



Step 8: Configure UAA

1. Select **UAA**.

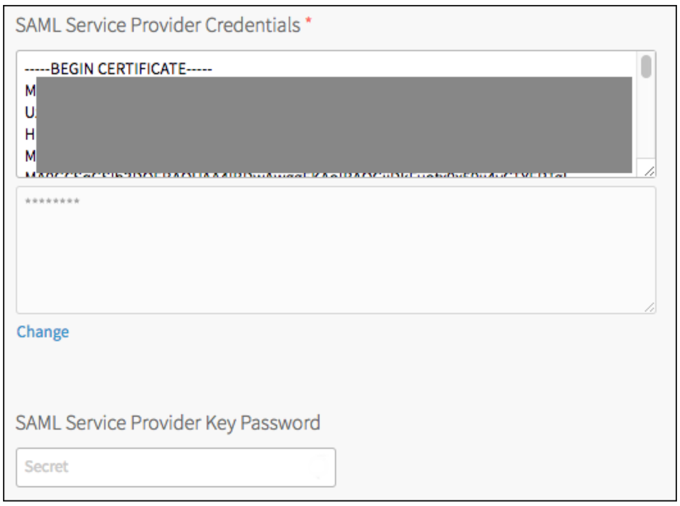
2. (Optional) Under **JWT Issuer URI**, enter the URI that UAA uses as the issuer when generating tokens.



3. Under **SAML Service Provider Credentials**, enter a certificate and private key to be used by UAA as a SAML Service Provider for signing outgoing SAML authentication requests. You can provide an existing certificate and private key from your trusted Certificate Authority or generate a self-signed certificate. The following domain must be associated with the certificate: `*.login.YOUR-SYSTEM-DOMAIN`.

> **Note**: The Pivotal Single Sign-On Service and Pivotal Spring Cloud Services tiles require the `*.login.YOUR-SYSTEM-DOMAIN`.

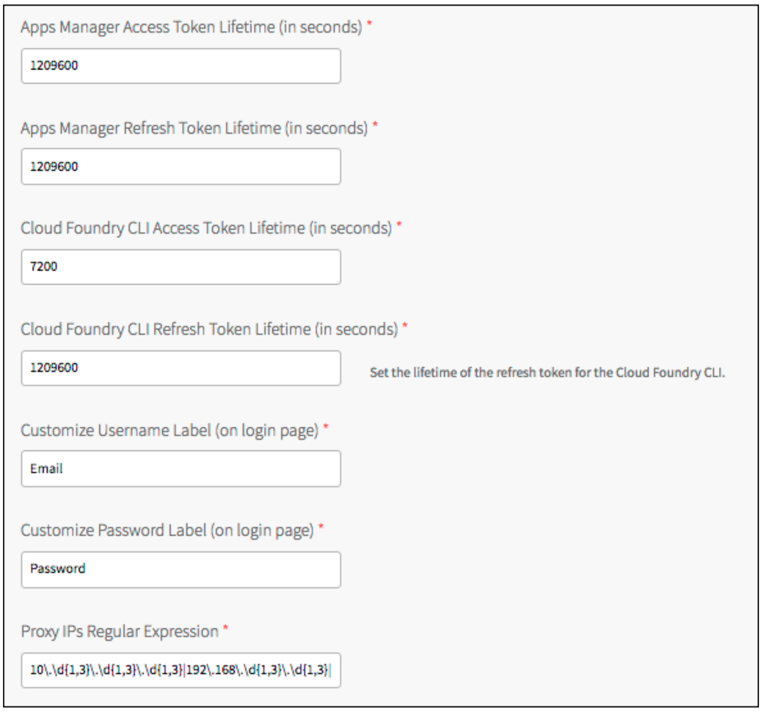
4. If the private key specified under **Service Provider Credentials** is password-protected, enter the password under **SAML Service Provider Key**



Password.

5. For **Signature Algorithm**, choose an algorithm from the dropdown menu to use for signed requests and assertions. The default value is `SHA256`.

6. (Optional) In the **Apps Manager Access Token Lifetime**, **Apps Manager Refresh Token Lifetime**, **Cloud Foundry CLI Access Token Lifetime**, and **Cloud Foundry CLI Refresh Token Lifetime** fields, change the lifetimes of tokens granted for Apps Manager and Cloud Foundry Command Line Interface (cf CLI) login access and refresh. Most deployments use the defaults.



7. (Optional) Customize the text prompts used for username and password from the cf CLI and Apps Manager login popup by entering values for **Customize Username Label (on login page)** and **Customize Password Label (on login page)**.

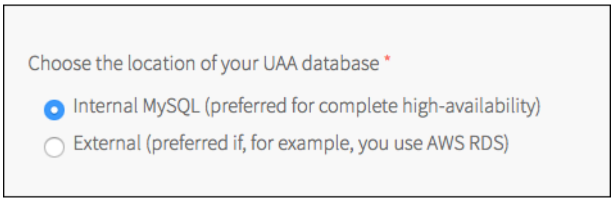
8. (Optional) The **Proxy IPs Regular Expression** field contains a pipe-delimited set of regular expressions that UAA considers to be reverse proxy IP addresses. UAA respects the `x-forwarded-for` and `x-forwarded-proto` headers coming from IP addresses that match these regular expressions. To configure UAA to respond properly to Gorouter or HAProxy requests coming from a public IP address, append a regular expression or regular expressions to match the public IP address.

9. You can configure UAA to use the internal MySQL database provided with PCF, or you can configure an external database provider. Follow the procedures in either the Internal Database Configuration or the External Database Configuration section below.

> **Note**: If you are performing an upgrade, do not modify your existing internal database configuration or you may lose data. You must migrate your existing data before changing the configuration. See Upgrading Pivotal Cloud Foundry for additional upgrade information, and contact Pivotal Support for help.

Internal Database Configuration

1. Select **Internal MySQL**.

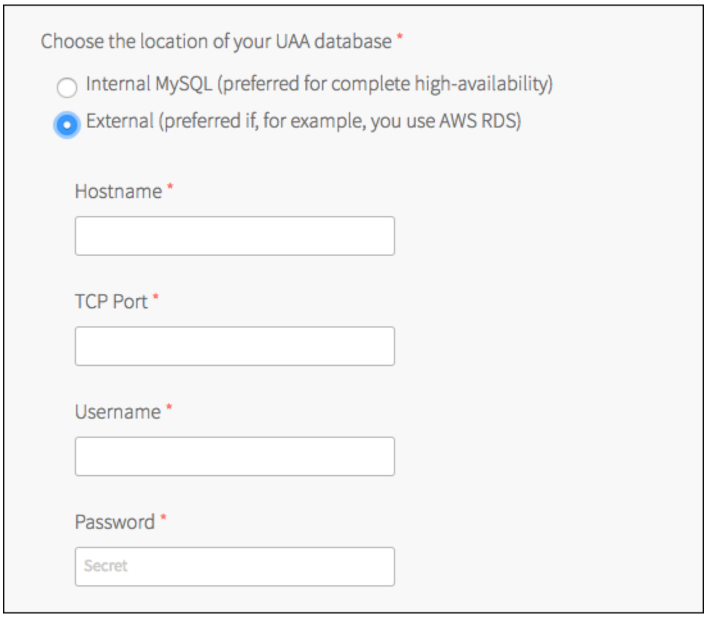


2. Click **Save**.

3. Ensure that you complete the “Configure Internal MySQL” step later in this topic to configure high availability and automatic backups for your internal MySQL databases.

External Database Configuration

1. From the **UAA** section in Pivotal Application Service (PAS), select **External**.



2. For **Hostname**, enter the hostname of the database server.

3. For **TCP Port**, enter the port of the database server.

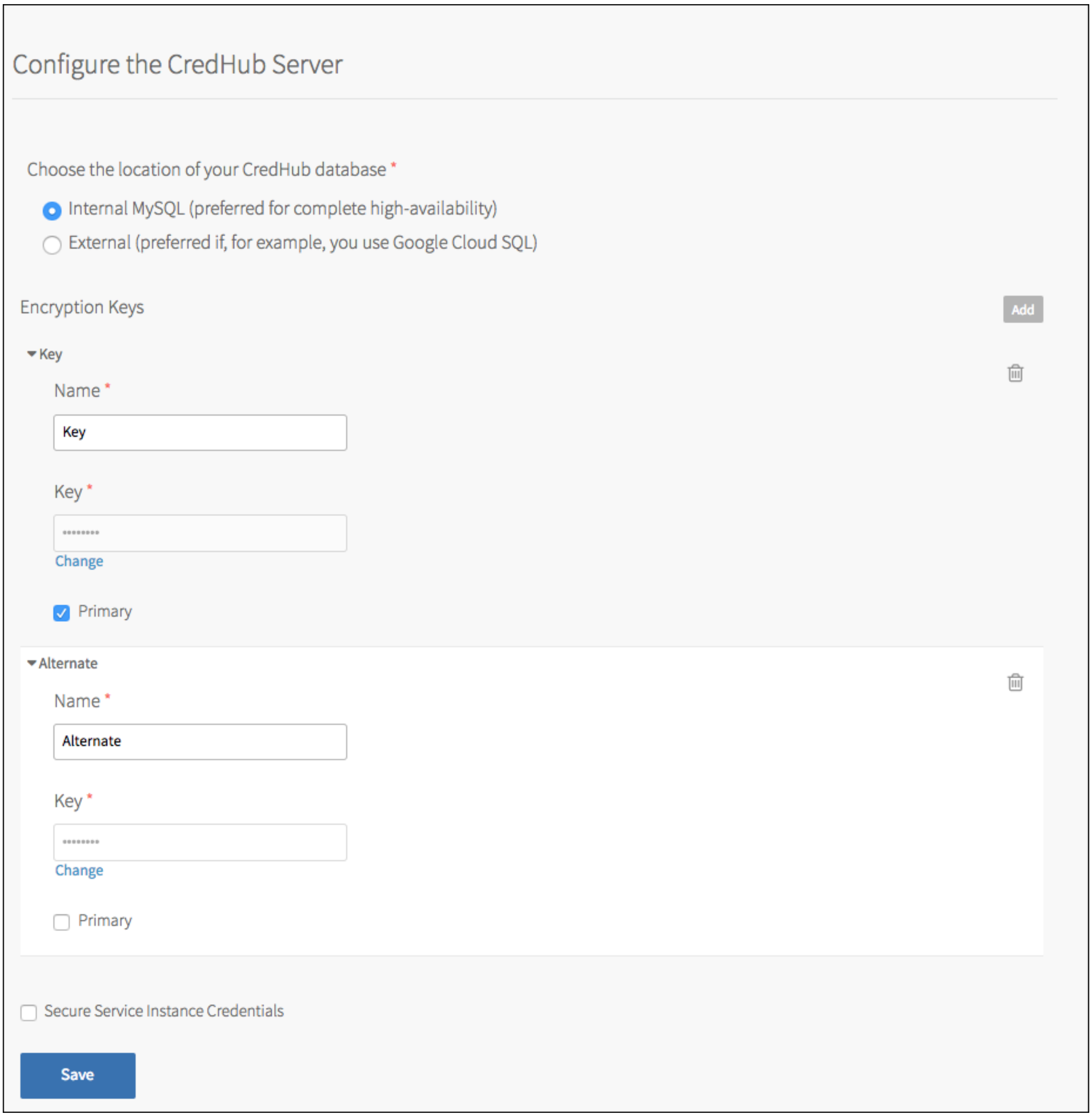
4. For **User Account and Authentication database username**, specify a unique username that can access this specific database on the database server.

5. For **User Account and Authentication database password**, specify a password for the provided username.

6. Click **Save**.

Step 9: (Optional) Configure CredHub

1. Select **Credhub**.



2. Choose the location of your CredHub Database. PAS includes this CredHub database for services to store their service instance credentials.
    a. If you chose **External**, enter the following:
        - **Hostname**: The IP address of your database server.
        - **TCP Port**: The port of your database server, such as `3306`.
        - **Username**: A unique username that can access this specific database on the database server.
        - **Password**: The password for the provided username.
        - **Database CA Certificate**: Enter a certificate to use for encrypting traffic to and from the database.
3. Under **Encryption Keys**, specify a key to use for encrypting and decrypting the values stored in the CredHub database.
    - **Name**: Enter the name of the key.
    - **Key**: Enter a key that is at least 20 characters in length.
    - **Primary**: Select this checkbox to use this key as your primary key.

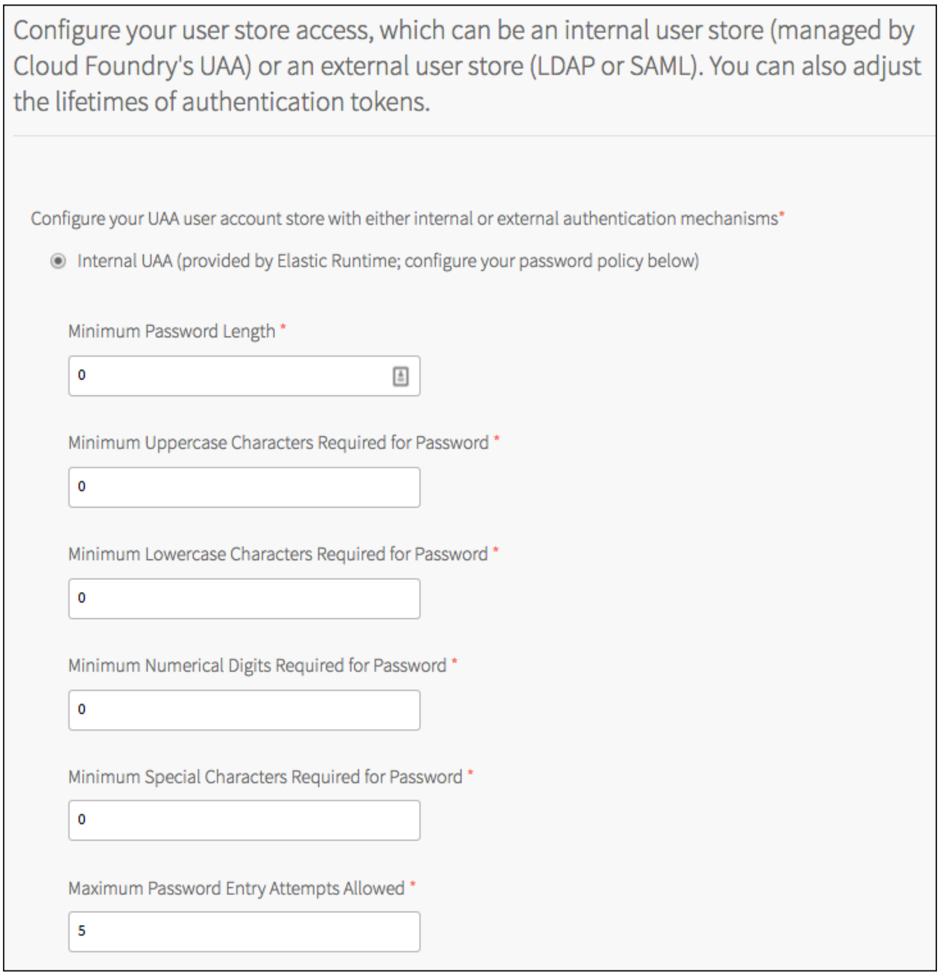
    > **Note**: Ensure that you only mark one key as **Primary**. The UI includes an **Add** button to add more keys to support key rotation. For more information, see the Rotating PAS CredHub Encryption Keys topic.

4. If your deployment uses any PCF services that support storing service instance credentials in CredHub and you want to enable this feature, select the **Secure Service Instance Credentials** checkbox.
5. Select the **Resource Config** pane.
6. Under the **Job** column of the **CredHub** row, set the number of instances to `2`. This is the minimum instance count required for high availability.

Note: To use the Runtime CredHub feature, you must follow the additional steps in Securing Service Instance Credentials with Runtime CredHub.

Step 10: Configure Authentication and Enterprise SSO

1. Select **Authentication and Enterprise SSO**.



2. To authenticate user sign-ons, your deployment can use one of three types of user database: the UAA server's internal user store, an external SAML identity provider, or an external LDAP server.
   - To use the internal UAA, select the **Internal** option and follow the instructions in the Configuring UAA Password Policy topic to configure your password policy.
   - To connect to an external identity provider through SAML, scroll down to select the **SAML Identity Provider** option and follow the instructions in the Configuring PCF for SAML section of the *Configuring Authentication and Enterprise SSO for Pivotal Application Service (PAS)* topic.
   - To connect to an external LDAP server, scroll down to select the **LDAP Server** option and follow the instructions in the Configuring LDAP section of the *Configuring Authentication and Enterprise SSO for PAS* topic.

3. Click **Save**.

Step 11: Configure System Databases

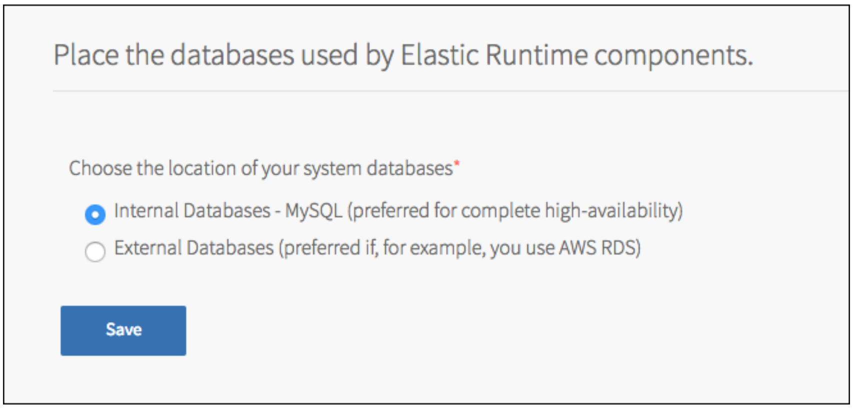
You can configure PAS to use the internal MySQL database provided with PCF, or you can configure an external database provider for the databases required by PAS.

Note: If you are performing an upgrade, do not modify your existing internal database configuration or you may lose data. You must migrate your existing data first before changing the configuration. Contact Pivotal Support for help. See Upgrading Pivotal Cloud Foundry for additional upgrade information.

Internal Database Configuration

If you want to use internal databases for your deployment, perform the following steps:

1. Select **Databases**.



2. Select **Internal Databases - MySQL**.
3. Click **Save**.

Then proceed to Step 12: (Optional) Configure Internal MySQL to configure high availability and automatic backups for your internal MySQL databases.

External Database Configuration

Note: To configure an external database for UAA, see the *External Database Configuration* section of Configure UAA.

Note: The exact procedure to create databases depends upon the database provider you select for your deployment. The following procedure uses AWS RDS as an example. You can configure a different database provider that provides MySQL support, such as Google Cloud SQL.

Warning: Protect whichever database you use in your deployment with a password.

To create your Pivotal Application Service (PAS) databases, perform the following steps:

1. Add the `ubuntu` account key pair from your IaaS deployment to your local SSH profile so you can access the Ops Manager VM. For example, in AWS, you add a key pair created in AWS:

```
$ ssh-add aws-keypair.pem
```

2. SSH in to your Ops Manager using the Ops Manager FQDN and the username `ubuntu`:

```
$ ssh ubuntu@OPS-MANAGER-FQDN
```

3. Log in to your MySQL database instance using the appropriate hostname and user login values configured in your IaaS account. For example, to log in to your AWS RDS instance, run the following MySQL command:

```
$ mysql --host=RDSHOSTNAME --user=RDSUSERNAME --password=RDSPASSWORD
```

4. Run the following MySQL commands to create databases for the eleven PAS components that require a relational database:

```
CREATE database ccdb;
CREATE database notifications;
CREATE database autoscale;
CREATE database app_usage_service;
CREATE database routing;
CREATE database diego;
CREATE database account;
CREATE database nfsvolume;
CREATE database networkpolicyserver;
CREATE database silk;
CREATE database locket;
```

5. Type `exit` to quit the MySQL client, and `exit` again to close your connection to the Ops Manager VM.

6. In PAS, select **Databases**.

7. Select the **External Databases** option.

8. For **Hostname**, enter the hostname of the database server.

9. For **TCP Port**, enter the port of the database server.

10. Each component that requires a relational database has two corresponding fields: one for the database username and one for the database password. For each set of fields, specify a unique username that can access this specific database on the database server and a password for the provided username.

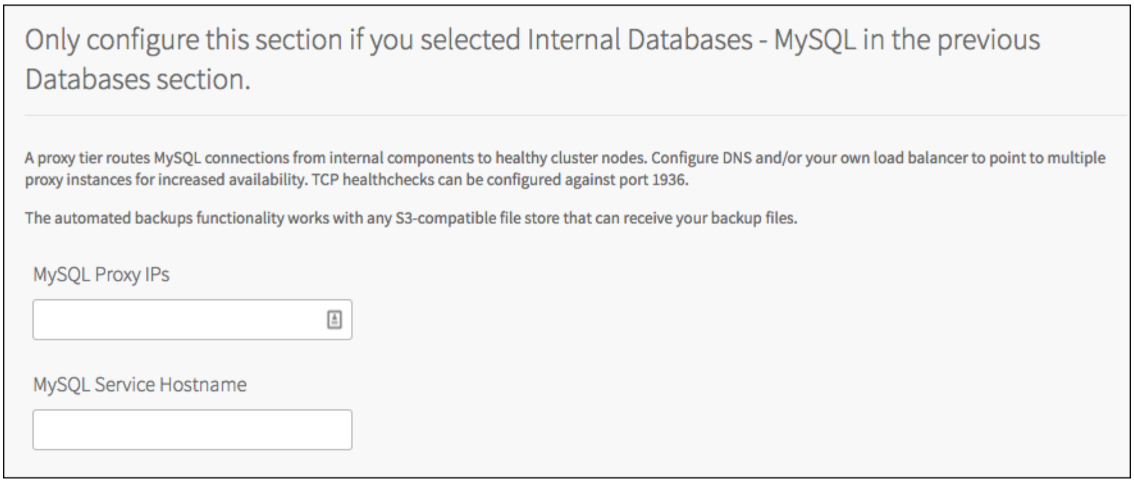
11. Click **Save**.

Step 12: (Optional) Configure Internal MySQL

Note: You only need to configure this section if you have selected **Internal Databases - MySQL** in the **Databases** section.

1. Select **Internal MySQL**.

2. In the **MySQL Proxy IPs** field, enter one or more comma-delimited IP addresses that are not in the reserved CIDR range of your network. If a MySQL node fails, these proxies re-route connections to a healthy node. See the MySQL Proxy topic for more information.



3. For **MySQL Service Hostname**, enter an IP address or hostname for your load balancer. If a MySQL proxy fails, the load balancer re-routes connections to a healthy proxy. If you leave this field blank, components are configured with the IP address of the first proxy instance entered above.

   Warning: You must configure a load balancer to achieve complete high-availability.

4. In the **Replication canary time period** field, leave the default of 30 seconds or modify the value based on the needs of your deployment. Lower numbers cause the canary to run more frequently, which means that the canary reacts more quickly to replication failure but adds load to the database.

5. In the **Replication canary read delay** field, leave the default of 20 seconds or modify the value based on the needs of your deployment. This field configures how long the canary waits, in seconds, before verifying that data is replicating across each MySQL node. Clusters under heavy load can experience a small replication lag as write-sets are committed across the nodes.

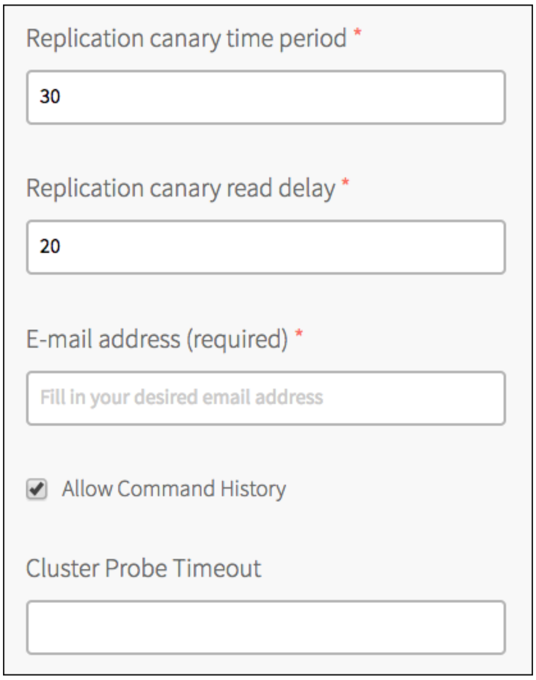
6. (**Required**): In the **E-mail address** field, enter the email address where the MySQL service sends alerts when the cluster experiences a replication issue or when a node is not allowed to auto-rejoin the cluster.

7. To prohibit the creation of command line history files on the MySQL nodes, disable the **Allow Command History** checkbox.

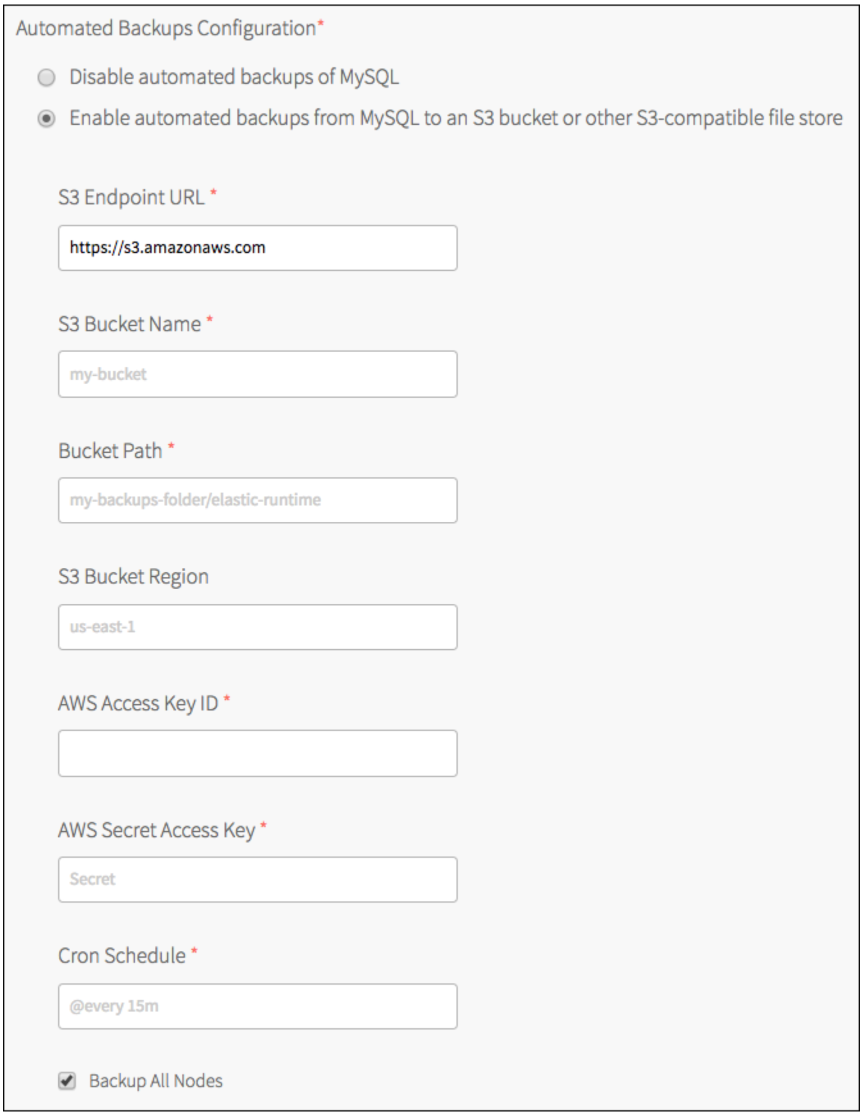
8. To allow the admin and roadmin to connect from any remote host, enable the **Allow Remote Admin Access** checkbox. When the checkbox is disabled, admins must `bosh ssh` into each MySQL VM to connect as the MySQL super user.

> **Note:** Network configuration and Application Security Groups restrictions may still limit a client's ability to establish a connection with the databases.

9. For **Cluster Probe Timeout**, enter the maximum amount of time, in seconds, that a new node will search for existing cluster nodes. If left blank, the default value is 10 seconds.



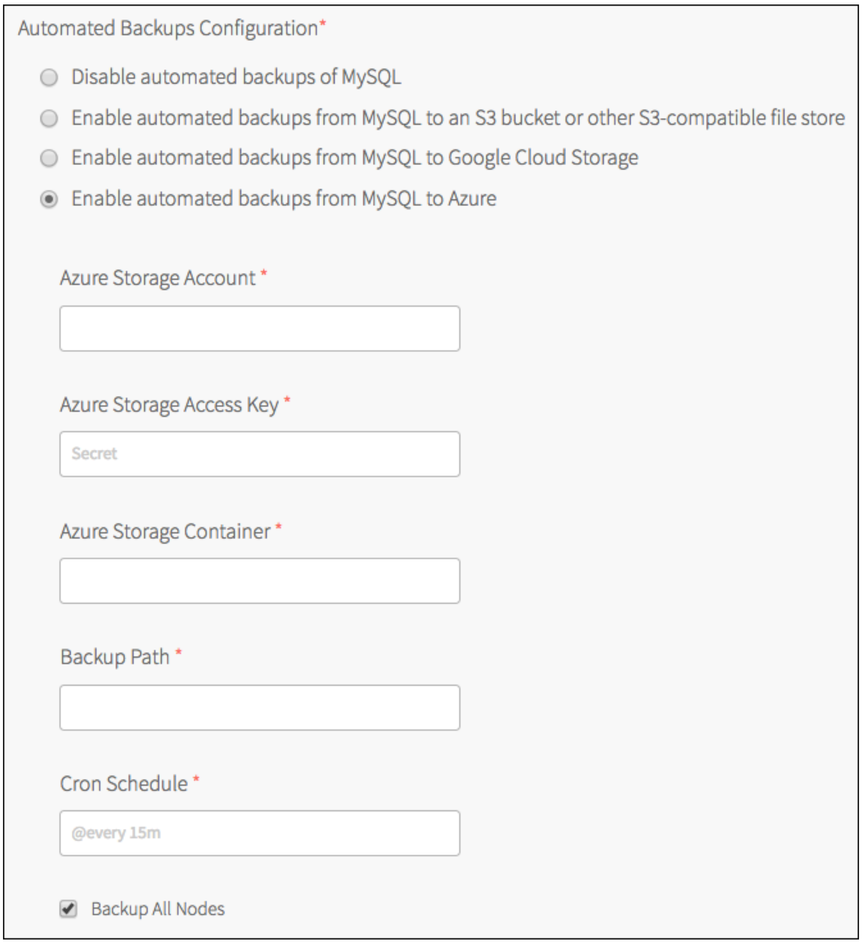
10. Under **Automated Backups Configuration**, choose one of three options for MySQL backups:
    - **Disable automatic backups of MySQL** disables automatic backups, but you can still deploy the Backup Prepare Node if you use BOSH Backup and Restore to back up your MySQL database. For more information, see the Backing Up Pivotal Cloud Foundry with BBR topic.
    - **Enable automated backups from MySQL to an S3 bucket or other S3-compatible file store** saves your backups to an existing Amazon Web Services (AWS) or Ceph S3-compatible blobstore.



This option requires the following fields:

- For **S3 Bucket Name**, enter the name of your S3 bucket. Do not include an `s3://` prefix, a trailing `/`, or underscores. If the bucket does not already exist, it will be created automatically.
- For **Bucket Path**, specify a folder within the bucket to hold your MySQL backups. Do not include a trailing `/`.
- For **S3 Bucket Region**, enter the AWS region where the bucket is located, such as `us-east-1`.
- For **AWS Access Key ID** and **AWS Secret Access Key**, enter your AWS or Ceph credentials.
- For **Cron Schedule**, enter a valid cron expression to schedule your automated backups. Cron uses your computer's local time zone.
- Enable **Backup All Nodes** to make unique backups from each instance of the MySQL server rather than just the first MySQL server instance.

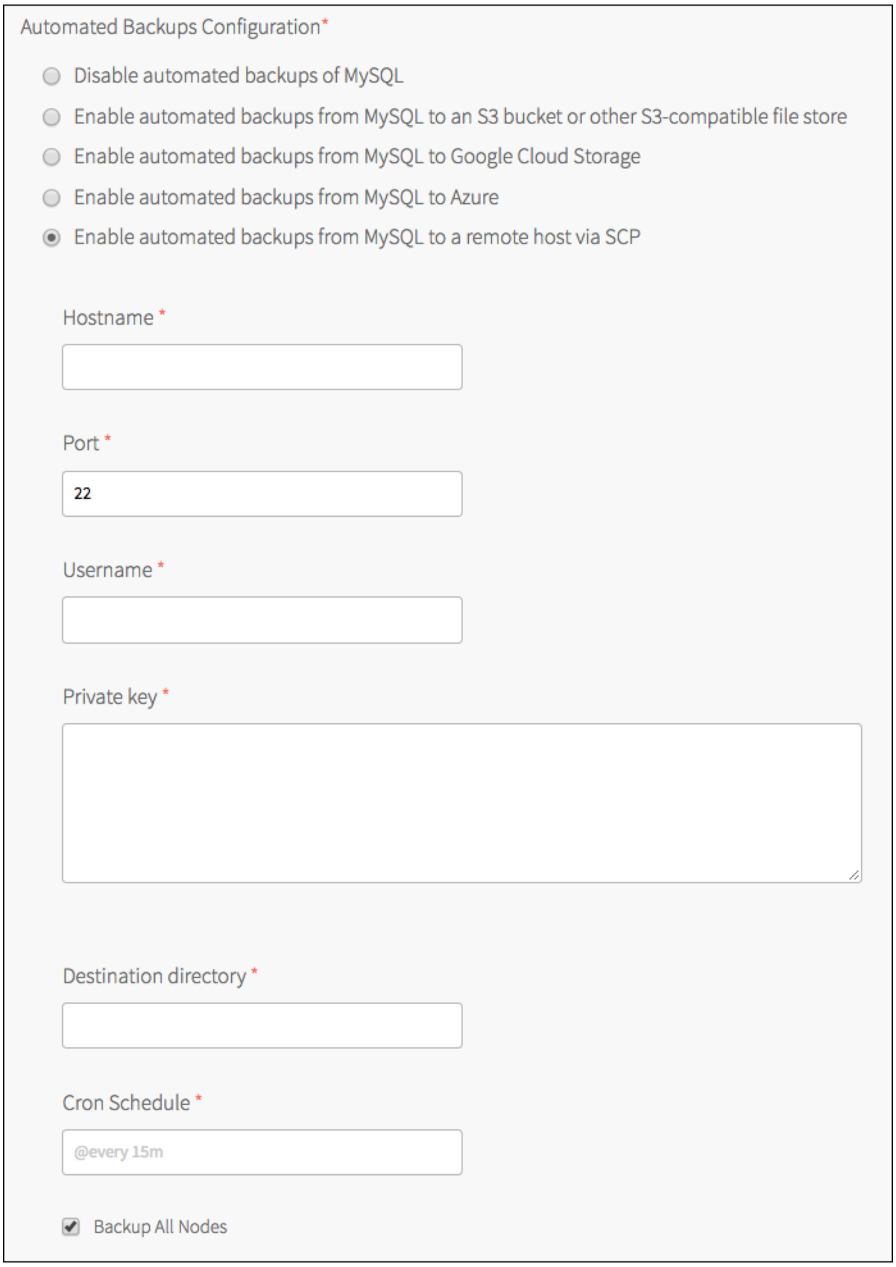
- **Enable automated backups from MySQL to Azure** saves your backups to Azure.



This option requires the following fields:

- For **Azure Storage Account**, enter the name of an existing Azure storage account where backups will be uploaded. For more information about creating and managing an Azure storage account, see the Azure documentation.
- For **Azure Storage Access Key**, enter an Azure storage access key for the storage account.
- For **Azure Storage Container**, enter the name of an existing Azure storage container that will store the backups.
- For **Backup Path**, enter the path within the Azure storage container where backups will be uploaded.
- For **Cron Schedule**, enter a valid cron expression to schedule your automated backups. Cron uses your computer's local time zone.
- Enable **Backup All Nodes** to make unique backups from each instance of the MySQL server rather than just the first MySQL server instance.

- **Enable automated backups from MySQL to a remote host via SCP**saves your backups to a remote host using secure copy protocol (SCP).



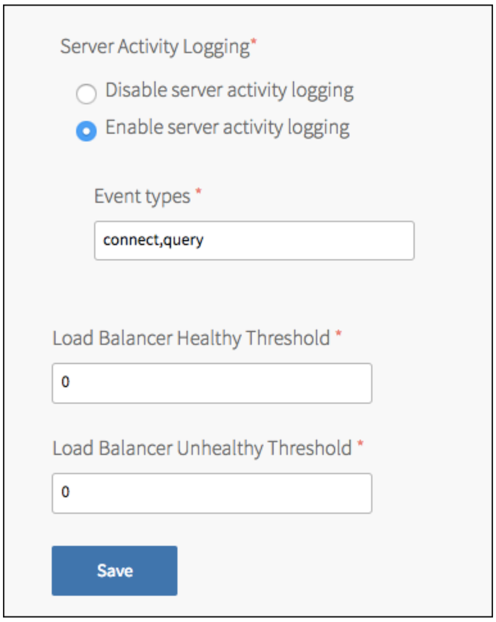
This option requires the following fields:

- For **Hostname**, enter the name of your SCP host.
- For **Port**, enter your SCP port. This should be the TCP port that your SCP host uses for SSH. The default port is `22`.
- For **Username**, enter your SSH username for the SCP host.
- For **Private key**, paste in your SSH private key.
- For **Destination directory**, enter the directory on the SCP host where you want to save backup files.
- For **Cron Schedule**, enter a valid cron expression to schedule your automated backups. Cron uses your computer's local time zone.
- Enable **Backup All Nodes** to make unique backups from each instance of the MySQL server rather than just the first MySQL server instance.

> **Note**: If you choose to enable automated MySQL backups, set the number of instances for the **Backup Prepare Node** under the **Resource Config** section of the Pivotal Application Service (PAS) tile to `1`.

11. If you want to log audit events for internal MySQL, select **Enable server activity logging** under **Server Activity Logging**.

    a. For the **Event types** field, you can enter the events you want the MySQL service to log. By default, this field includes `connect` and `query`, which tracks who connects to the system and what queries are processed. For more information, see the Logging Events section of the MariaDB documentation.



12. Enter values for the following fields:
    - **Load Balancer Healthy Threshold**: Specifies the amount of time, in seconds, to wait until declaring the MySQL proxy instance started. This allows an external load balancer time to register the instance as healthy.
    - **Load Balancer Unhealthy Threshold**: Specifies the amount of time, in seconds, that the MySQL proxy continues to accept connections before shutting down. During this period, the healthcheck reports as unhealthy to cause load balancers to fail over to other proxies. You must enter a value greater than or equal to the maximum time it takes your load balancer to consider a proxy instance unhealthy, given repeated failed healthchecks.

13. If you want to enable the MySQL interruptor feature, select the checkbox to **Prevent node auto re-join**. This feature stops all writes to the MySQL database if it notices an inconsistency in the dataset between the nodes. For more information, see the Interruptor section in the MySQL for PCF documentation.

14. Click **Save**.

Step 13: Configure File Storage

For production-level PCF deployments on OpenStack, the recommended selection is **External S3-Compatible**.

For more factors to consider when selecting file storage, see Considerations for Selecting File Storage in Pivotal Cloud Foundry.

Internal Filestore

Internal file storage is only appropriate for small, non-production deployments.

To use the PCF internal filestore, perform the following steps:

1. In the Pivotal Application Service (PAS) tile, select **File Storage**.

2. Select **Internal WebDAV**, and click **Save**.

External S3 or Ceph Filestore

To use an external S3-compatible filestore for your Pivotal Application Service (PAS) file storage, perform the following steps:

1. In the PAS tile, select **File Storage**.

2. Select the **External S3-Compatible Filestore** option and complete the following fields:
    - Prepend `https://` to the endpoint for your region and copy it into the Ops Manager **URL Endpoint** field. For example, in the **us-west-2** region, use `https://s3-us-west-2.amazonaws.com/`.

- Enter the **Access Key** and **Secret Key** of the `pcf-user` you created when configuring AWS for PCF.
- For **S3 Signature Version** and **Region**, use the **V4 Signature** values. AWS recommends using Signature Version 4.
- For **Region**, enter the region in which your S3 buckets are located, such as `us-west-2`.
- Select **Server-side Encryption (available for AWS S3 only)** to encrypt the contents of your S3 filestore.
- Enter values for the remaining fields as follows:

Ops Manager Field	Value	Description
Buildpacks Bucket Name	`pcf-buildpacks-bucket`	This S3 bucket stores app buildpacks.
Droplets Bucket Name	`pcf-droplets-bucket`	This S3 bucket stores app droplets. Pivotal recommends that you use a unique bucket name for droplets, but you can also use the same name as above.
Packages Bucket Name	`pcf-packages-bucket`	This S3 bucket stores app packages. Pivotal recommends that you use a unique bucket name for packages, but you can also use the same name as above.
Resources Bucket Name	`pcf-resources-bucket`	This S3 bucket stores app resources. Pivotal recommends that you use a unique bucket name for app resources, but you can also use the same name as above.

3. Click **Save**.

Note: For more information regarding AWS S3 Signatures, see the Authenticating Requests topic in the AWS documentation.

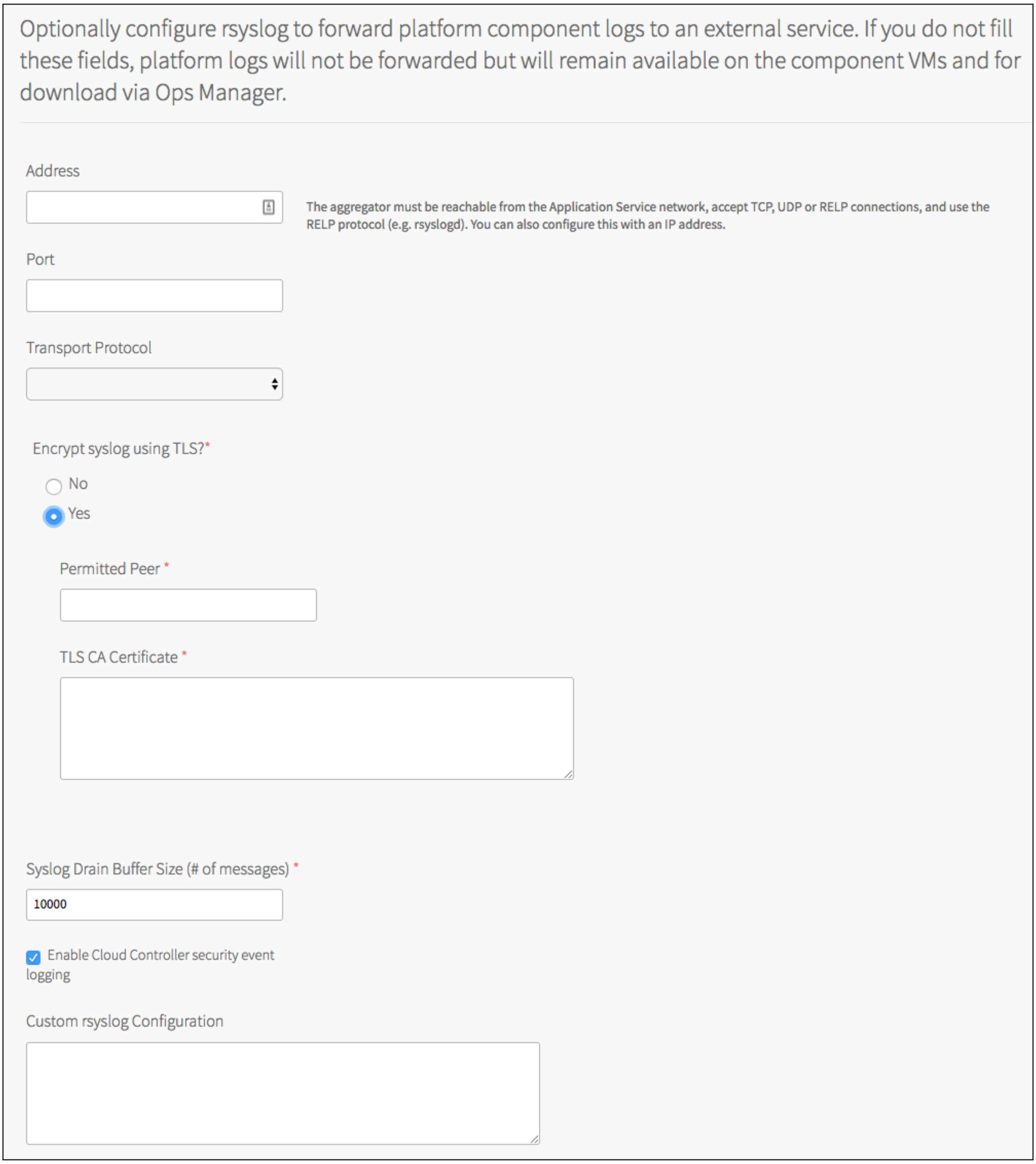
Other IaaS Storage Options

Google Cloud Storage and Azure Storage are also available as file storage options but have not been evaluated for typical PCF on OpenStack installations.

Step 14: (Optional) Configure System Logging

If you forward logging messages to an external Reliable Event Logging Protocol (RELP) server, complete the following steps:

1. Select the **System Logging** section that is located within your PAS **Settings** tab.



2. Enter the IP address of your syslog server in **Address**.

3. Enter the port of your syslog server in **Port**. The default port for a syslog server is `514`.

> **Note**: The host must be reachable from the PAS network, accept TCP connections, and use the RELP protocol. Ensure your syslog server listens on external interfaces.

4. Select a **Transport Protocol** to use when forwarding logs.

5. If you plan to use TLS encryption when sending logs to the remote server, select **Yes** when answering the **Encrypt syslog using TLS?** question.

    a. In the **Permitted Peer** field, enter either the name or SHA1 fingerprint of the remote peer.
    b. In the **TLS CA Certificate** field, enter the TLS CA Certificate for the remote server.

6. For the **Syslog Drain Buffer Size**, enter the number of messages the Doppler server can hold from Metron agents before the server starts to drop them. See the Loggregator Guide for Cloud Foundry Operators topic for more details.

7. If you want to include security events in your log stream, select the **Enable Cloud Controller security event logging** checkbox. This logs all API requests, including the endpoint, user, source IP address, and request result, in the Common Event Format (CEF).

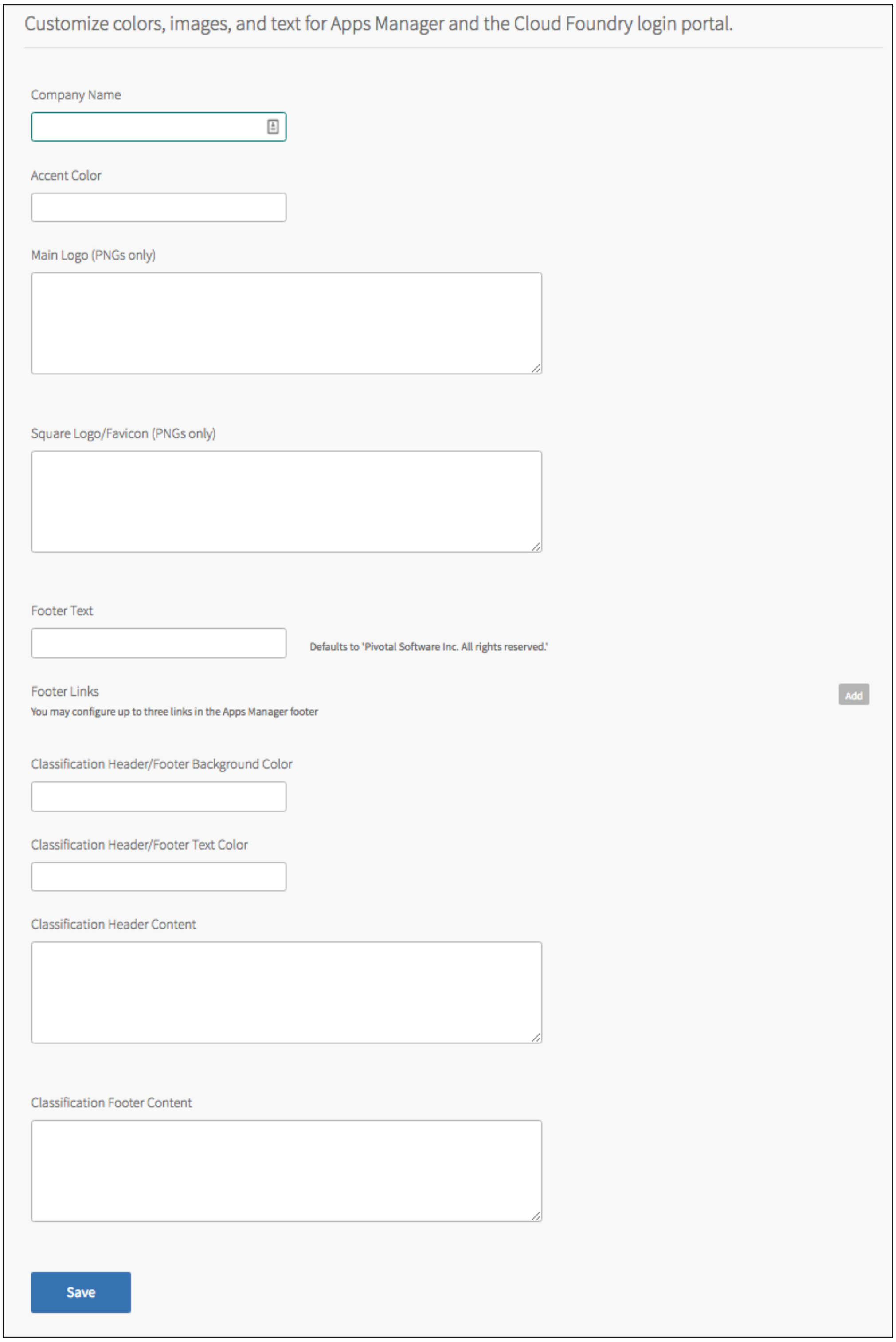
8. If you want to specify a custom syslog formatting rule, enter it in the **Custom rsyslog Configuration** field in RainerScript syntax.

9. Click **Save**.

Step 15: (Optional) Customize Apps Manager

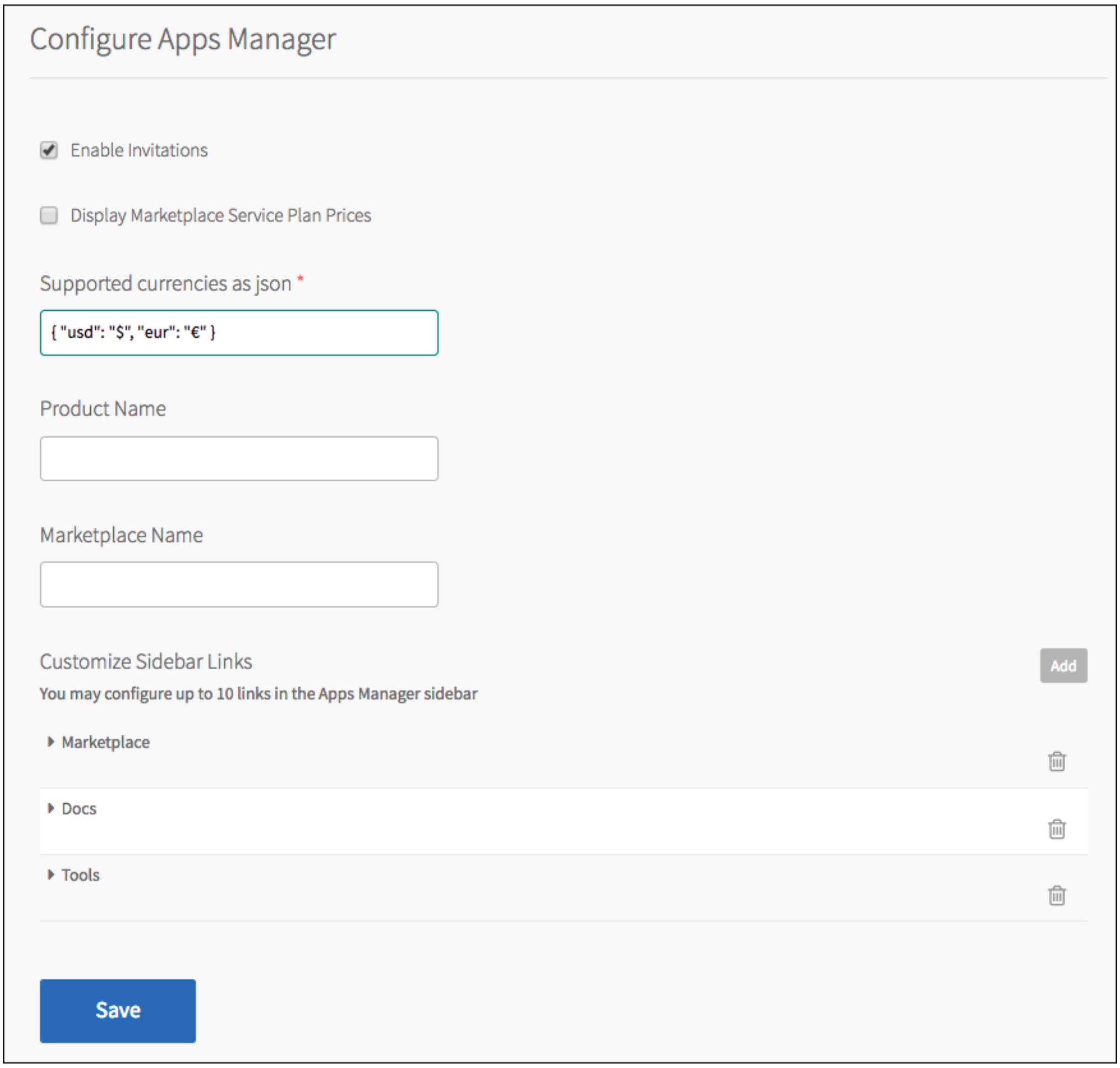
The **Custom Branding** and **Apps Manager** sections customize the appearance and functionality of Apps Manager. Refer to Custom Branding Apps Manager for descriptions of the fields on these pages and for more information about customizing Apps Manager.

1. Select **Custom Branding**. Use this section to configure the text, colors, and images of the interface that developers see when they log in, create an account, reset their password, or use Apps Manager.



2. Click **Save** to save your settings in this section.

3. Select **Apps Manager**.



4. Select **Enable Invitations** to enable invitations in Apps Manager. Space Managers can invite new users for a given space, Org Managers can invite new users for a given org, and Admins can invite new users across all orgs and spaces. See the [Inviting New Users]() section of the *Managing User Roles with Apps Manager* topic for more information.

5. Select **Display Marketplace Service Plan Prices** to display the prices for your services plans in the Marketplace.

6. Enter the **Supported currencies as json** to appear in the Marketplace. Use the format `{"CURRENCY-CODE":"SYMBOL"}`. This defaults to `{"usd": "$", "eur": "€"}`.

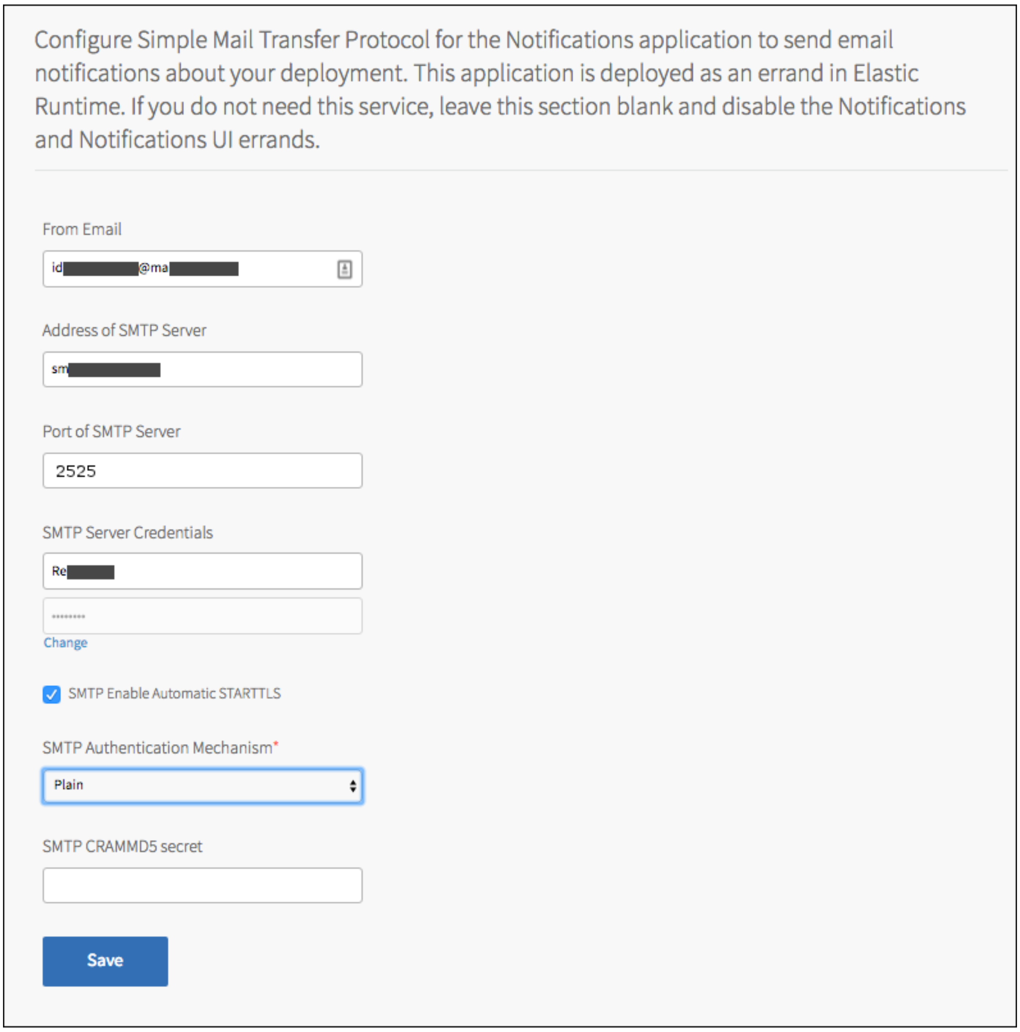
7. Use **Product Name**, **Marketplace Name**, and **Customize Sidebar Links** to configure page names and sidebar links in the **Apps Manager** and **Marketplace** pages.

8. Click **Save** to save your settings in this section.

Step 16: (Optional) Configure Email Notifications

PAS uses SMTP to send invitations and confirmations to Apps Manager users. You must complete the **Email Notifications** page if you want to enable end-user self-registration.

1. Select **Email Notifications**.

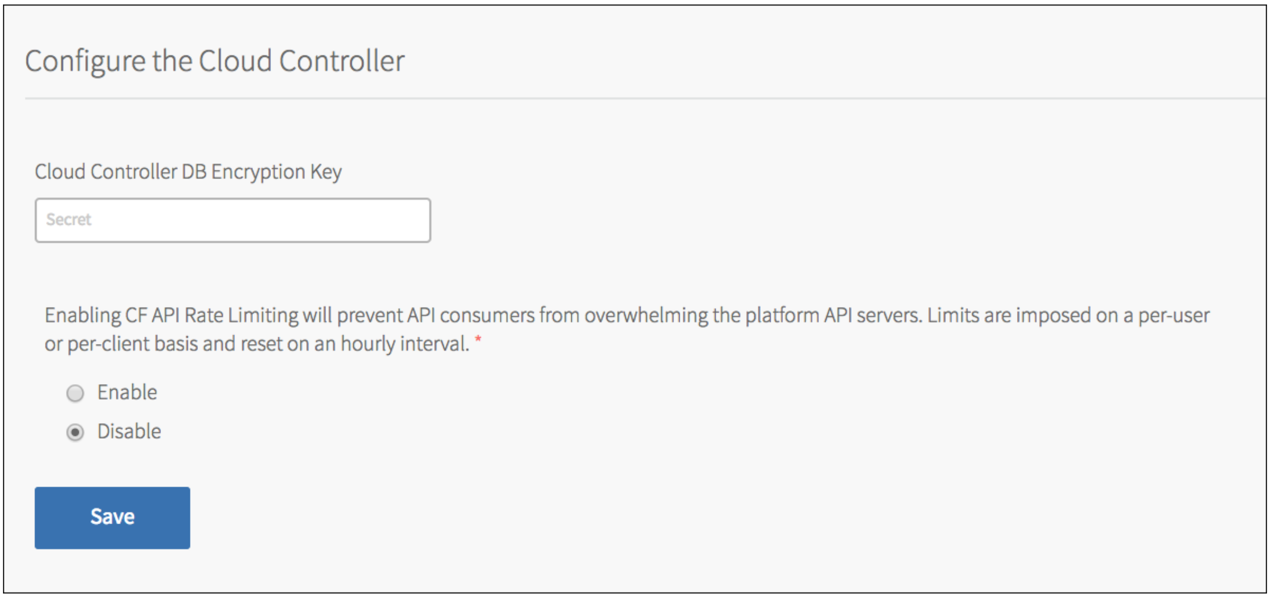


2. Enter your reply-to and SMTP email information.
3. For **SMTP Authentication Mechanism**, select `none`.
4. Click **Save**.

> **Note**: If you do not configure the SMTP settings using this form, the administrator must create orgs and users using the cf CLI tool. See Creating and Managing Users with the cf CLI for more information.

Step 17: Configure Cloud Controller

1. Click **Cloud Controller**.



2. Enter your **Cloud Controller DB Encryption Key** if all of the following are true:
   - You deployed Pivotal Application Service (PAS) previously.
   - You then stopped PAS or it crashed.
   - You are re-deploying PAS with a backup of your Cloud Controller database.

   See Backing Up Pivotal Cloud Foundry for more information.

3. CF API Rate Limiting prevents API consumers from overwhelming the platform API servers. Limits are imposed on a per-user or per-client basis and reset on an hourly interval.

   To disable CF API Rate Limiting, select **Disable** under **Enable CF API Rate Limiting**. To enable CF API Rate Limiting, perform the following steps:

   a. Under **Enable CF API Rate Limiting**, select **Enable**.
   b. For **General Limit**, enter the number of requests a user or client is allowed to make over an hour interval for all endpoints that do not have a custom limit. The default value is `2000`.
   c. For **Unauthenticated Limit**, enter the number of requests an unauthenticated client is allowed to make over an hour interval. The default value is `100`.

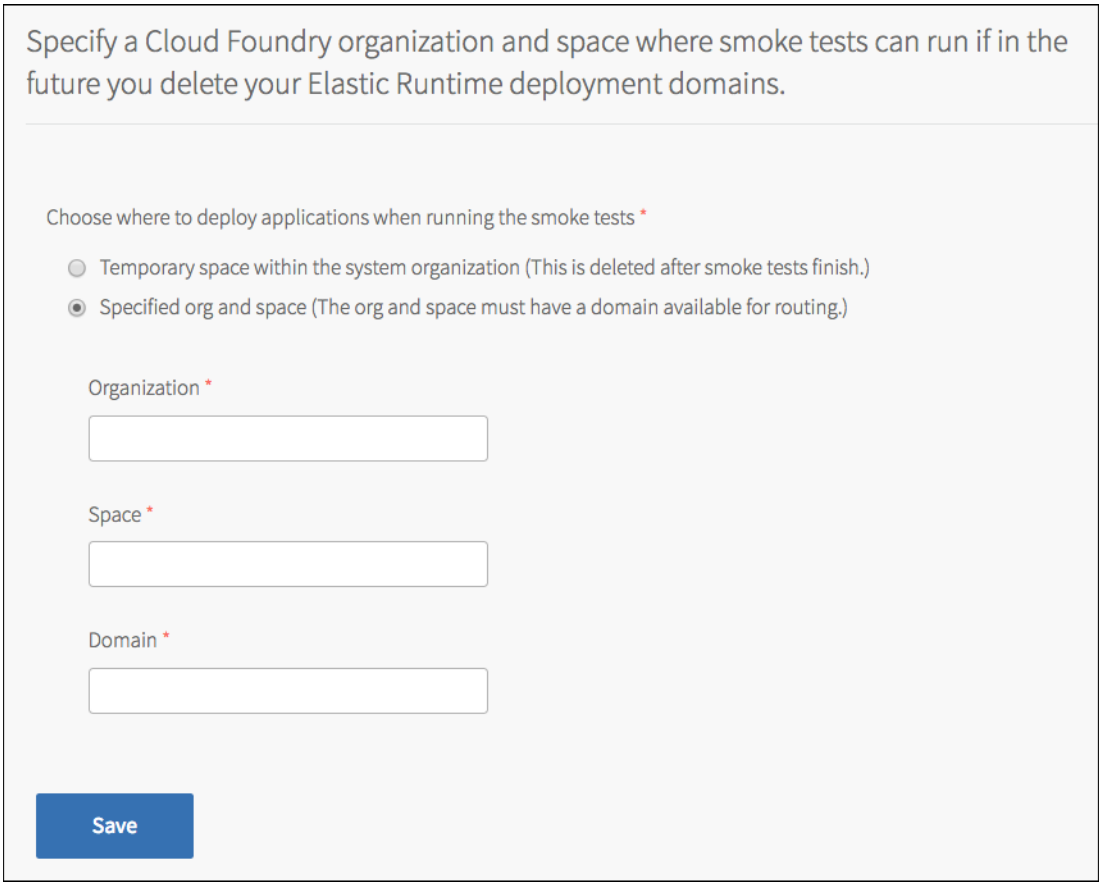
4. Click **Save**.

Step 18: Configure Smoke Tests

The Smoke Tests errand runs basic functionality tests against your Pivotal Application Service (PAS) deployment after an installation or update. In this section, choose where to run smoke tests. In the **Errands** section, you can choose whether or not to run the Smoke Tests errand.

1. Select **Smoke Tests**.

2. If you have a shared apps domain, select **Temporary space within the system organization**, which creates a temporary space within the `system` organization for running smoke tests and deletes the space afterwards. Otherwise, select **Specified org and space** and complete the fields to specify where you want to run smoke tests.



3. Click **Save**.

Step 19: (Optional) Enable Advanced Features

The **Advanced Features** section of Pivotal Application Service (PAS) includes new functionality that may have certain constraints. Although these features are fully supported, Pivotal recommends caution when using them in production environments.

Diego Cell Memory and Disk Overcommit

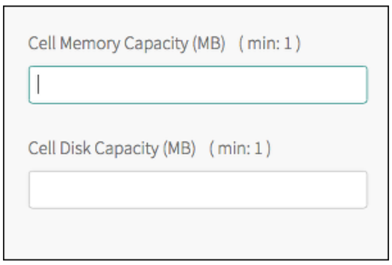
If your apps do not use the full allocation of disk space and memory set in the **Resource Config** tab, you might want use this feature. These fields control the amount to overcommit disk and memory resources to each Diego Cell VM.

For example, you might want to use the overcommit if your apps use a small amount of disk and memory capacity compared to the amounts set in the **Resource Config** settings for **Diego Cell**.

> **Note**: Due to the risk of app failure and the deployment-specific nature of disk and memory use, Pivotal has no recommendation about how much, if any, memory or disk space to overcommit.

To enable overcommit, follow these steps:

1. Select **Advanced Features**.



2. Enter the total desired amount of Diego cell memory value in the **Cell Memory Capacity (MB)** field. Refer to the **Diego Cell** row in the **Resource Config** tab for the current Cell memory capacity settings that this field overrides.

3. Enter the total desired amount of Diego cell disk capacity value in the **Cell Disk Capacity (MB)** field. Refer to the **Diego Cell** row in the **Resource Config** tab for the current Cell disk capacity settings that this field overrides.

4. Click **Save**.

> **Note**: Entries made to each of these two fields set the total amount of resources allocated, not the overage.

Whitelist for Non-RFC-1918 Private Networks

Some private networks require extra configuration so that internal file storage (WebDAV) can communicate with other PCF processes.

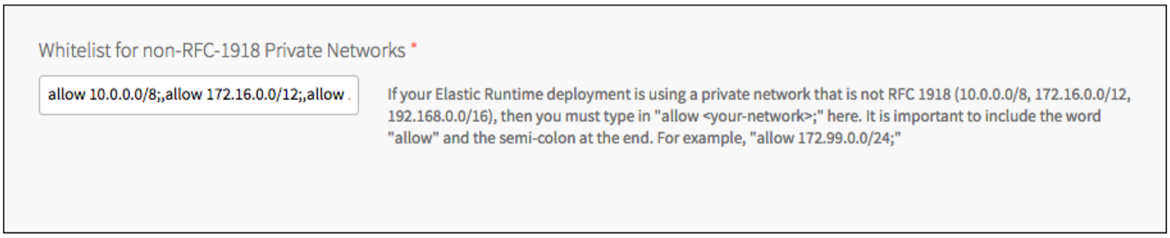
The **Whitelist for non-RFC-1918 Private Networks** field is provided for deployments that use a non-RFC 1918 private network. This is typically a private network other than `10.0.0.0/8`, `172.16.0.0/12`, or `192.168.0.0/16`.

Most PCF deployments do not require any modifications to this field.

To add your private network to the whitelist, perform the following steps:

1. Select **Advanced Features**.

2. Append a new `allow` rule to the existing contents of the **Whitelist for non-RFC-1918 Private Networks** field.



Include the word `allow`, the network CIDR range to allow, and a semi-colon (`;`) at the end. For example: `allow 172.99.0.0/24;`

3. Click **Save**.

CF CLI Connection Timeout

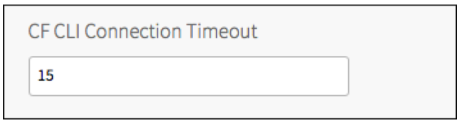
The **CF CLI Connection Timeout** field allows you to override the default five second timeout of the Cloud Foundry Command Line Interface (cf CLI) used within your PCF deployment. This timeout affects the cf CLI command used to push PAS errand apps such as Notifications, Autoscaler, and Apps Manager.

Set the value of this field to a higher value, in seconds, if you are experiencing domain name resolution timeouts when pushing errands in PAS.

To modify the value of the **CF CLI Connection Timeout**, perform the following steps:

1. Select **Advanced Features**.

2. Add a value, in seconds, to the **CF CLI Connection Timeout** field.



3. Click **Save**.

Step 20: Configure Errands

Errands are scripts that Ops Manager runs automatically when it installs or uninstalls a product, such as a new version of Pivotal Application Service (PAS). There are two types of errands: *post-deploy errands* run after the product is installed, and *pre-delete errands* run before the product in uninstalled.

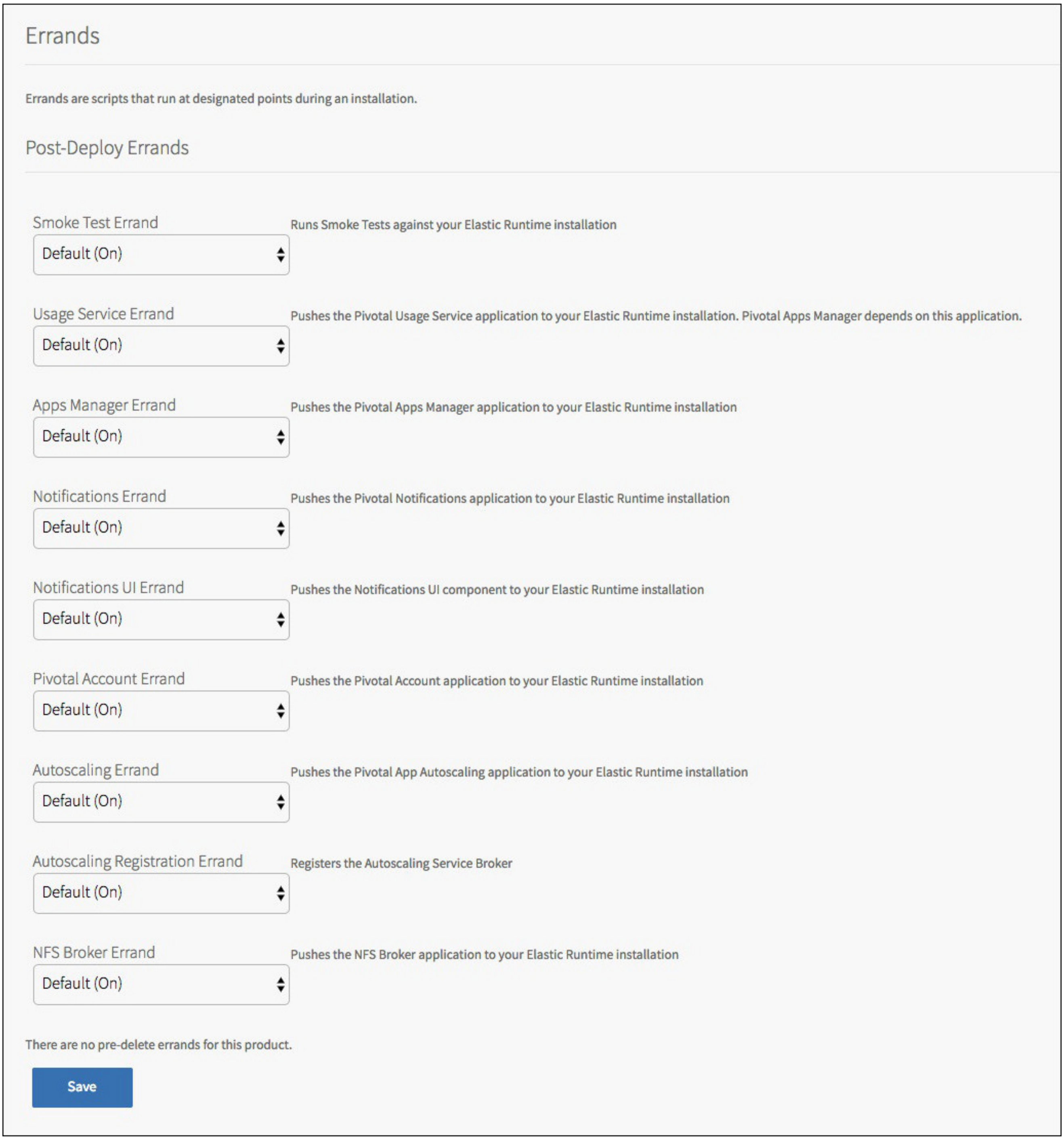
By default, Ops Manager always runs pre-delete errands, and only runs post-deploy errands when the product has changed since the last time Ops

Manager installed something. In PAS, the Smoke Test Errand defaults to always run.

The PAS tile **Errands** pane lets you change these run rules. For each errand, you can select **On** to run it always, **Off** to never run it, or **When Changed** to run it only when the product has changed since the last install.

For more information about how Ops Manager manages errands, see the Managing Errands in Ops Manager topic.

> **Note**: Several errands deploy apps that provide services for your deployment, such as Autoscaling and Notifications. Once one of these apps is running, selecting **Off** for the corresponding errand on a subsequent installation does not stop the app.



- **Smoke Test Errand** verifies that your deployment can do the following:
  - Push, scale, and delete apps
  - Create and delete orgs and spaces
- **Usage Service Errand** deploys the Pivotal Usage Service application, which Apps Manager depends on.
- **Apps Manager Errand** deploys Apps Manager, a dashboard for managing apps, services, orgs, users, and spaces. Until you deploy Apps Manager, you

must perform these functions through the cf CLI. After Apps Manager has been deployed, Pivotal recommends deselecting the checkbox for this errand on subsequent PAS deployments. For more information about Apps Manager, see the Getting Started with the Apps Manager topic.

- **Notifications Errand** deploys an API for sending email notifications to your PCF platform users.

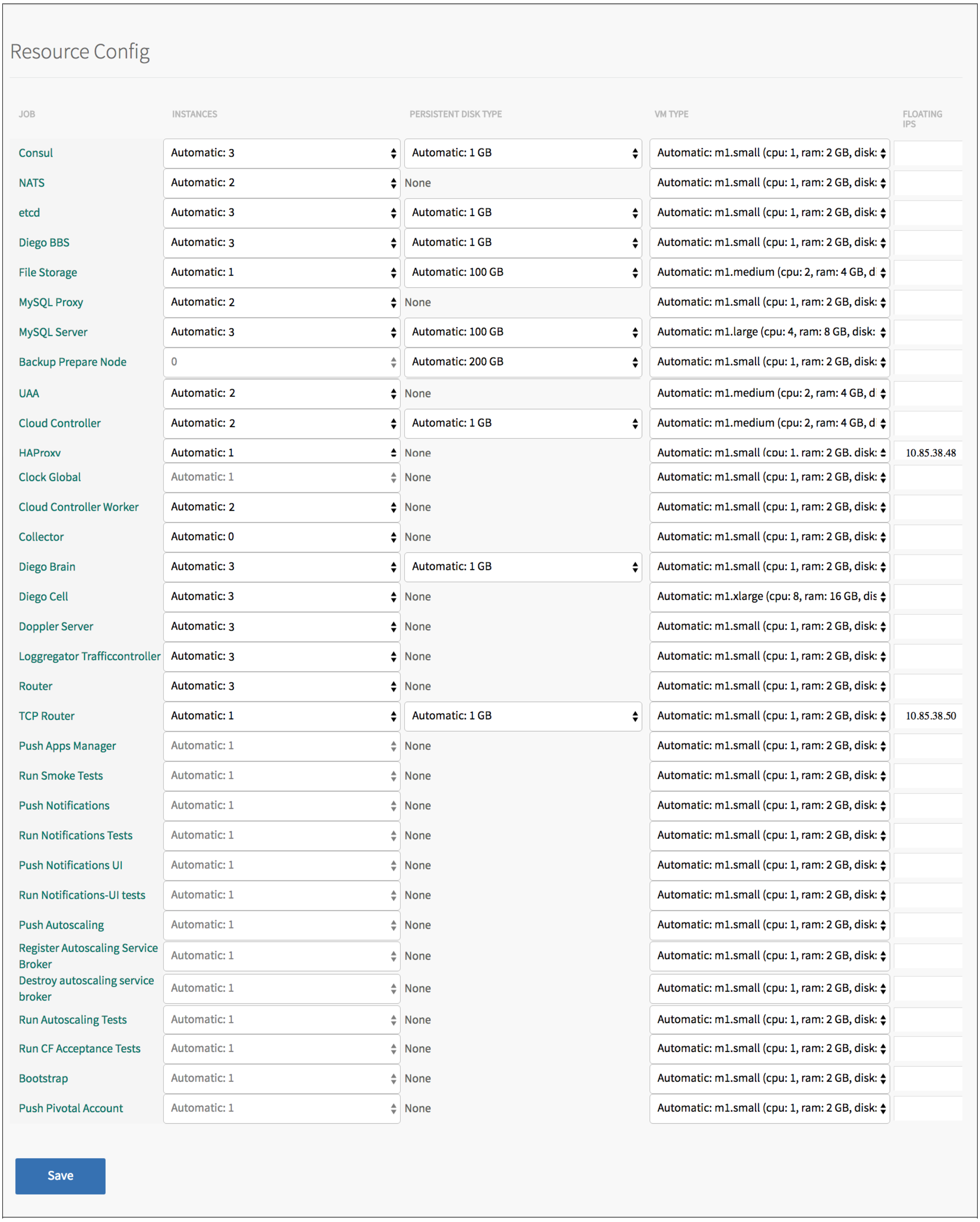
> **Note**: The Notifications app requires that you configure SMTP with a username and password, even if you set the value of **SMTP Authentication Mechanism** to `none`.

- **Notifications UI Errand** deploys a dashboard for users to manage notification subscriptions.
- **Pivotal Account Errand** deploys Pivotal Account, a dashboard that allows users to create and manage their accounts. In the Pivotal Account dashboard, users can launch applications, manage their profiles, manage account security, manage notifications, and manage approvals. See the Enabling Pivotal Account topic for more information.
- **Autoscaling Errand** enables you to configure your apps to automatically scale in response to changes in their usage load. See the Scaling an Application Using Autoscaler topic for more information.
- **Autoscaling Registration Errand** makes the Autoscaling service available to your applications. Without this errand, you cannot bind the Autoscaling app to your apps.
- **NFS Broker Errand** enables you to use NFS Volume Services by installing the NFS Broker app in PAS. See the Enabling NFS Volume Services topic for more information.

Step 21: Enable Traffic to Private Subnet

Unless you are using your own load balancer, you must enable traffic flow to the OpenStack private subnet as follows. Give each HAProxy a way of routing traffic into the private subnet by providing public IP addresses as floating IP addresses.

1. Click **Resource Config**.



Resource Config

JOB	INSTANCES	PERSISTENT DISK TYPE	VM TYPE	FLOATING IPS
Consul	Automatic: 3	Automatic: 1 GB	Automatic: m1.small (cpu: 1, ram: 2 GB, disk:	
NATS	Automatic: 2	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk:	
etcd	Automatic: 3	Automatic: 1 GB	Automatic: m1.small (cpu: 1, ram: 2 GB, disk:	
Diego BBS	Automatic: 3	Automatic: 1 GB	Automatic: m1.small (cpu: 1, ram: 2 GB, disk:	
File Storage	Automatic: 1	Automatic: 100 GB	Automatic: m1.medium (cpu: 2, ram: 4 GB, d	
MySQL Proxy	Automatic: 2	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk:	
MySQL Server	Automatic: 3	Automatic: 100 GB	Automatic: m1.large (cpu: 4, ram: 8 GB, disk:	
Backup Prepare Node	0	Automatic: 200 GB	Automatic: m1.small (cpu: 1, ram: 2 GB, disk:	
UAA	Automatic: 2	None	Automatic: m1.medium (cpu: 2, ram: 4 GB, d	
Cloud Controller	Automatic: 2	Automatic: 1 GB	Automatic: m1.medium (cpu: 2, ram: 4 GB, d	
HAProxy	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk:	10.85.38.48
Clock Global	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk:	
Cloud Controller Worker	Automatic: 2	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk:	
Collector	Automatic: 0	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk:	
Diego Brain	Automatic: 3	Automatic: 1 GB	Automatic: m1.small (cpu: 1, ram: 2 GB, disk:	
Diego Cell	Automatic: 3	None	Automatic: m1.xlarge (cpu: 8, ram: 16 GB, dis	
Doppler Server	Automatic: 3	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk:	
Loggregator Trafficcontroller	Automatic: 3	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk:	
Router	Automatic: 3	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk:	
TCP Router	Automatic: 1	Automatic: 1 GB	Automatic: m1.small (cpu: 1, ram: 2 GB, disk:	10.85.38.50
Push Apps Manager	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk:	
Run Smoke Tests	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk:	
Push Notifications	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk:	
Run Notifications Tests	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk:	
Push Notifications UI	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk:	
Run Notifications-UI tests	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk:	
Push Autoscaling	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk:	
Register Autoscaling Service Broker	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk:	
Destroy autoscaling service broker	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk:	
Run Autoscaling Tests	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk:	
Run CF Acceptance Tests	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk:	
Bootstrap	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk:	
Push Pivotal Account	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk:	

Save

2. Enter one or more IP addresses in **Floating IPs** for each HAProxy.

3. (Optional) If you have enabled the TCP Routing feature, enter one or more IP addresses in **Floating IPs** column for each TCP Router.

4. Click **Save**.

Step 22: (Optional) Scale Down and Disable Resources

> 💡 **Note**: The **Resource Config** pane has fewer VMs if you are installing the [Small Footprint Runtime](#).

> 💡 **Note**: The Small Footprint Runtime does not default to a highly available configuration. It defaults to the minimum configuration. If you want to make the Small Footprint Runtime highly available, scale the **Compute**, **Router**, and **Database** VMs to `3` instances and scale the **Control** VM to `2` instances.

Pivotal Application Service (PAS) defaults to a highly available resource configuration. However, you may need to perform additional procedures to make your deployment highly available. See the [Zero Downtime Deployment and Scaling in CF](#) and the [Scaling Instances in PAS](#) topics for more information.

If you do not want a highly available resource configuration, you must scale down your instances manually by navigating to the **Resource Config** section and using the drop-down menus under **Instances** for each job.

By default, PAS also uses an internal filestore and internal databases. If you configure PAS to use external resources, you can disable the corresponding system-provided resources in Ops Manager to reduce costs and administrative overhead.

Complete the following procedures to disable specific VMs in Ops Manager:

1. Click **Resource Config**.

2. If you configure PAS to use an external S3-compatible filestore, edit the following fields:
   - **File Storage**: Enter `0` in **Instances**.

3. If you selected **External** when configuring the UAA or System databases, edit the following fields:
   - **MySQL Proxy**: Enter `0` in **Instances**.
   - **MySQL Server**: Enter `0` in **Instances**.
   - **MySQL Monitor**: Enter `0` in **Instances**.
   - **Cloud Controller Database**: Enter `0` in **Instances**.
   - **UAA Database**: Enter `0` in **Instances**.

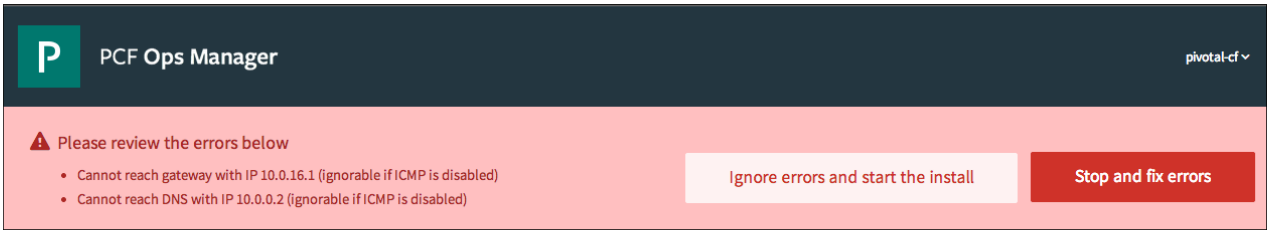
4. If you are not using HAProxy, enter `0` in the **Instances** field for **HAProxy**.

5. Click **Save**.

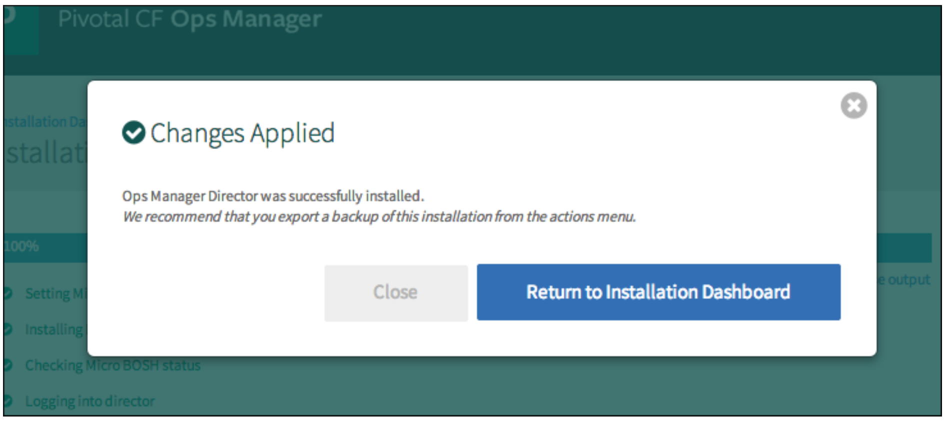
Step 23: Complete PAS Installation

1. Click the **Installation Dashboard** link to return to the Installation Dashboard.

2. Click **Apply Changes**. If the following ICMP error message appears, click **Ignore errors and start the install**.



3. PAS installs. The image shows the **Changes Applied** message that Ops Manager displays when the installation process successfully completes.



Return to Installing Pivotal Cloud Foundry on OpenStack.

PCF on vSphere Requirements

Page last updated:

This guide describes how to install Pivotal Cloud Foundry (PCF) on vSphere.

If you experience a problem while following the steps below, refer to the Known Issues topics or to Diagnosing Problems in PCF.

General Requirements

The following are general requirements for deploying and managing a PCF deployment with Ops Manager and Pivotal Application Service (PAS):

- A wildcard DNS record that points to your router or load balancer. Alternatively, you can use a service such as xip.io. For example, `203.0.113.0.xip.io`.

  PAS gives each application its own hostname in your app domain. With a wildcard DNS record, every hostname in your domain resolves to the IP address of your router or load balancer, and you do not need to configure an A record for each app hostname. For example, if you create a DNS record `*.example.com` pointing to your load balancer or router, every application deployed to the `example.com` domain resolves to the IP address of your router.
- At least one wildcard TLS certificate that matches the DNS record you set up above, `*.example.com`.
- Sufficient IP allocation:
  - One static IP address for either HAProxy or one of your gorouters
  - One IP address for each VM instance
  - An additional IP address for each compilation worker
    So the formula for total IPs needed is `IPs needed = static IPs + VM instances + compilation workers`

  > **Note**: Pivotal recommends that you allocate at least 36 dynamic IP addresses when deploying Ops Manager and PAS. BOSH requires additional dynamic IP addresses during installation to compile and deploy VMs, install PAS, and connect to services.

- One or more NTP servers if not already provided by your IaaS.
- (**Recommended**) A network without DHCP available for deploying the PAS VMs.

  > **Note**: If you have DHCP, refer to the Troubleshooting Guide to avoid issues with your installation.

- (**Optional**) External storage. When you deploy PCF, you can select internal file storage or external file storage, either network-accessible or IaaS-provided, as an option in the PAS tile. Pivotal recommends using external storage whenever possible. See Upgrade Considerations for Selecting File Storage in Pivotal Cloud Foundry for a discussion of how file storage location affects platform performance and stability during upgrades.
- (**Optional**) External databases. When you deploy PCF, you can select internal or external databases for the BOSH Director and for PAS. Pivotal recommends using external databases in production deployments.
- (**Optional**) External user stores. When you deploy PCF, you can select a SAML user store for Ops Manager or a SAML or LDAP user store for PAS, to integrate existing user accounts.
- The most recent version of the Cloud Foundry Command Line Interface (cf CLI).

vSphere Requirements

> **Note**: If you are using the Cisco Nexus 1000v Switch, refer to the Using the Cisco Nexus 1000v Switch with Ops Manager topic for more information.

> **Note**: When installing Ops Manager on a vSphere environment with multiple ESXi hosts, you must use network-attached or shared storage devices. Local storage devices do not support sharing across multiple ESXi hosts.

The following are the minimum resource requirements for maintaining a Pivotal Cloud Foundry (PCF) deployment with Ops Manager and Pivotal Application Service (PAS) on vSphere:

- vSphere 6.5, 6.0, or 5.5
- Disk space: 2TB recommended

- Memory: 120GB
- Two public IP addresses: One for PAS and one for Ops Manager
- vCPU cores: 80
- Overall CPU: 28 GHz
- vSphere editions: standard and above
- Ops Manager must have HTTPS access to vCenter and ESX hosts on TCP port 443.
- A configured vSphere cluster:
  - If you enable vSphere DRS (Distributed Resource Scheduler) for the cluster, you must set the Automation level to **Partially automated** or **Fully automated**. If you set the Automation level to **Manual**, the BOSH automated installation will fail with a `power_on_vm` error when BOSH attempts to create virtual machines (VMs).
  - Disable hardware virtualization if your vSphere hosts do not support VT-X/EPT. If you are unsure whether the VM hosts support VT-x/EPT, you should disable this setting. If you leave this setting enabled and the VM hosts do not support VT-x/EPT, then each VM requires manual intervention in vCenter to continue powering on without the Intel virtualized VT-x/EPT. Refer to the vCenter help topic at Configuring Virtual Machines > Setting Virtual Processors and Memory > Set Advanced Processor Options for more information.

Note: If you are deploying PCF behind a firewall, see the Preparing Your Firewall for Deploying Pivotal Cloud Foundry topic.

Instance Number and Scaling Requirements

By default, PAS deploys the number of VM instances required to run a highly available configuration of PCF. If you are deploying a test or sandbox PCF that does not require HA, then you can scale down the number of instances in your deployment.

For information about the number of instances required to run a minimal, non-HA PCF deployment, see Scaling PAS.

vSphere Service Account Requirements

Ops Manager requires read/write permissions to the datacenter level of the vSphere Inventory Hierarchy to successfully install. Pivotal recommends defining a custom role for the service account that has all privileges for all objects in the datacenter, including propagating privileges to children.

For a complete list of the minimum required vSphere privileges, see the vSphere Service Account Requirements topic.

Since Ops Manager passes all required credentials through to BOSH, you only need one service account with the required vSphere privileges to complete the installation. Setting up separate service accounts for Ops Manager and BOSH is not necessary or recommended.

Note: You can also apply the default VMware Administrator System Role to the service account to achieve the appropriate permission level.

vSphere Security Documents

- vSphere Security guide (PDF)
  This guide contains best practices for securing and managing a vSphere installation.

Step 1: Install Ops Manager

Complete the following procedures to install Ops Manager on vSphere:

1. Deploying Operations Manager to vSphere

2. Configuring Ops Manager Director for VMware vSphere

(Optional) Step 2: Install the IPsec Add-on

The PCF IPsec add-on secures network traffic within a PCF deployment and provides internal system protection if a malicious actor breaches your firewall. See the Securing Data in Transit with the IPsec Add-on topic for installation instructions.

> **Note**: You must install the PCF IPsec add-on before installing any other tiles to enable the IPsec functionality. Pivotal recommends installing IPsec immediately after Ops Manager, and before installing the PAS tile.

(Optional) Step 3: Install the NSX-T Tile

The NSX-T tile provides a single network for containers, VMs, and other entities. It enables NSX to enforce C2C policy and operators to configure additional policies through NSX Manager.

- You must install the NSX-T tile after you install or upgrade the Ops Manager Director tile.
- You must install the NSX-T tile before you install or upgrade the PAS tile.

See the NSX-T Container Plug-in for Kubernetes and Cloud Foundry - Installation and Administration Guide for more information.

Step 4: Install PAS

To install PAS on vSphere, perform the procedures in the Configuring PAS for vSphere topic.

Step 5: Create New User Accounts

Once you have successfully deployed PCF, add users to your account. Refer to the Creating New PAS User Accounts topic for more information.

Step 6: Target Your Deployment

Use the Cloud Foundry Command Line Interface (cf CLI) to target your deployment. If you have not installed the cf CLI, follow the instructions in Installing the cf CLI. For more information about the cf CLI, see Getting Started with the cf CLI.

> **Note**: To obtain the UAA admin name and password, refer to PAS **Credentials** in Ops Manager. You can also use the user that you created in Apps Manager, or create another user with the `create-user` command.

Additional Configuration

See the following topics for additional configuration information:

- Provisioning a Virtual Disk in vSphere
- Using the Cisco Nexus 1000v Switch with Ops Manager
- Using Ops Manager Resurrector on VMware vSphere
- Configuring SSL Termination for vSphere Deployments
- Understanding Availability Zones in VMware Installations
- Updating NSX Security Group and Load Balancer Information

vSphere Service Account Requirements

Page last updated:

This topic describes the minimum privileges required by the vSphere BOSH CPI. You must grant the following privileges to your vSphere service account to deploy Pivotal Cloud Foundry (PCF).

vCenter Root Privileges

Ops Manager assigns custom attributes to the virtual machines (VMs) it deploys to identify BOSH releases and job index information about each VM. vCenter APIs require root access to manage these custom attributes.

You must grant the following privileges on the root vCenter server entity to the service account:

Privilege (UI)	Privilege (API)
Read-only	System.Anonymous
	System.Read
	System.View
Manage custom attributes	Global.ManageCustomFields

vCenter Datacenter Privileges

You must grant the following privileges on any entities in a datacenter where you will deploy PCF:

Role Object

Privilege (UI)	Privilege (API)
Users inherit the Read-Only role from the vCenter root level	System.Anonymous
	System.Read
	System.View

Datastore Object

The following privileges must be set at the datacenter level to upload and delete virtual machine files.

Privilege (UI)	Privilege (API)
Allocate space	Datastore.AllocateSpace
Browse datastore	Datastore.Browse
Low level file operations	Datastore.FileManagement
Remove file	Datastore.DeleteFile
Update virtual machine files	Datastore.UpdateVirtualMachineFiles

Folder Object

Ops Manager creates a folder for VMs, stemcells, and persistent disks during installation. The folder contents change frequently as Ops Manager applies changes.

Privilege (UI)	Privilege (API)
Delete folder	Folder.Delete
Create folder	Folder.Create

Move folder	Folder.Move
Rename folder	Folder.Rename

Global Object

Privilege (UI)	Privilege (API)
Set custom attribute	Global.SetCustomField

Host Object

This setting allows BOSH to manage rules for Distributed Resource Scheduler (DRS) and VM affinity. BOSH requires this setting, but Ops Manager does not use this feature. See the BOSH documentation for more information.

Privilege (UI)	Privilege (API)
Modify cluster	Host.Inventory.EditCluster

Network Object

Privilege (UI)	Privilege (API)
Assign network	Network.Assign

Resource Object

When using `vAppImport` to clone a VM, BOSH requires the resource migration privileges to create a new, powered-off VM based on a given stemcell. BOSH migrates the VM to the destination datastore, where Ops Manager deploys the VM and powers it on.

Privilege (UI)	Privilege (API)
Assign virtual machine to resource pool	Resource.AssignVMToPool
Migrate powered off virtual machine	Resource.ColdMigrate
Migrate powered on virtual machine	Resource.HotMigrate

Virtual Machine Object

Configuration

Privilege (UI)	Privilege (API)
Add existing disk	VirtualMachine.Config.AddExistingDisk
Add new disk	VirtualMachine.Config.AddNewDisk
Add or remove device	VirtualMachine.Config.AddRemoveDevice
Advanced	VirtualMachine.Config.AdvancedConfig
Change CPU count	VirtualMachine.Config.CPUCount
Change resource	VirtualMachine.Config.Resource
Configure managedBy	VirtualMachine.Config.ManagedBy
Disk change tracking	VirtualMachine.Config.ChangeTracking
Disk lease	VirtualMachine.Config.DiskLease
Display connection settings	VirtualMachine.Config.MksControl
Extend virtual disk	VirtualMachine.Config.DiskExtend
Memory	VirtualMachine.Config.Memory

Modify device settings	VirtualMachine.Config.EditDevice
Raw device	VirtualMachine.Config.RawDevice
Reload from path	VirtualMachine.Config.ReloadFromPath
Remove disk	VirtualMachine.Config.RemoveDisk
Rename	VirtualMachine.Config.Rename
Reset guest information	VirtualMachine.Config.ResetGuestInfo
Set annotation	VirtualMachine.Config.Annotation
Settings	VirtualMachine.Config.Settings
Swapfile placement	VirtualMachine.Config.SwapPlacement
Unlock virtual machine	VirtualMachine.Config.Unlock

Guest Operations

Privilege (UI)	Privilege (API)
Guest Operation Program Execution	VirtualMachine.GuestOperations.Execute
Guest Operation Modifications	VirtualMachine.GuestOperations.Modify
Guest Operation Queries	VirtualMachine.GuestOperations.Query

Interaction

Privilege (UI)	Privilege (API)
Answer question	VirtualMachine.Interact.AnswerQuestion
Configure CD media	VirtualMachine.Interact.SetCDMedia
Console interaction	VirtualMachine.Interact.ConsoleInteract
Defragment all disks	VirtualMachine.Interact.DefragmentAllDisks
Device connection	VirtualMachine.Interact.DeviceConnection
Guest operating system management by VIX API	VirtualMachine.Interact.GuestControl
Power off	VirtualMachine.Interact.PowerOff
Power on	VirtualMachine.Interact.PowerOn
Reset	VirtualMachine.Interact.Reset
Suspend	VirtualMachine.Interact.Suspend
VMware Tools install	VirtualMachine.Interact.ToolsInstall

Inventory

Privilege (UI)	Privilege (API)
Create from existing	VirtualMachine.Inventory.CreateFromExisting
Create new	VirtualMachine.Inventory.Create
Move	VirtualMachine.Inventory.Move
Register	VirtualMachine.Inventory.Register
Remove	VirtualMachine.Inventory.Delete
Unregister	VirtualMachine.Inventory.Unregister

Provisioning

When cloning a stemcell, BOSH sets custom specifications, such as hostnames and network configurations, based on the stemcell operating system.

The VM download privilege allows BOSH to modify files within a VM, including links between VMs and persistent disks. When vMotion migrates disks in vSphere, BOSH uses these links to maintain the connections between VMs and their persistent disks.

Privilege (UI)	Privilege (API)
Allow disk access	VirtualMachine.Provisioning.DiskRandomAccess
Allow read-only disk access	VirtualMachine.Provisioning.DiskRandomRead
Allow virtual machine download	VirtualMachine.Provisioning.GetVmFiles
Allow virtual machine files upload	VirtualMachine.Provisioning.PutVmFiles
Clone template	VirtualMachine.Provisioning.CloneTemplate
Clone virtual machine	VirtualMachine.Provisioning.Clone
Customize	VirtualMachine.Provisioning.Customize
Deploy template	VirtualMachine.Provisioning.DeployTemplate
Mark as template	VirtualMachine.Provisioning.MarkAsTemplate
Mark as virtual machine	VirtualMachine.Provisioning.MarkAsVM
Modify customization specification	VirtualMachine.Provisioning.ModifyCustSpecs
Promote disks	VirtualMachine.Provisioning.PromoteDisks
Read customization specifications	VirtualMachine.Provisioning.ReadCustSpecs

Snapshot Management

Before Ops Manager deploys a new VM, it uses a snapshot to clone the stemcell image to the destination.

Privilege (UI)	Privilege (API)
Create snapshot	VirtualMachine.State.CreateSnapshot
Remove snapshot	VirtualMachine.State.RemoveSnapshot
Rename snapshot	VirtualMachine.State.RenameSnapshot
Revert snapshot	VirtualMachine.State.RevertToSnapshot

vApp Object

These privileges must be set at the resource pool level. `VApp.ApplicationConfig` is required when attaching or detaching persistent disks.

Privilege (UI)	Privilege (API)
Import	VApp.Import
vApp application configuration	VApp.ApplicationConfig

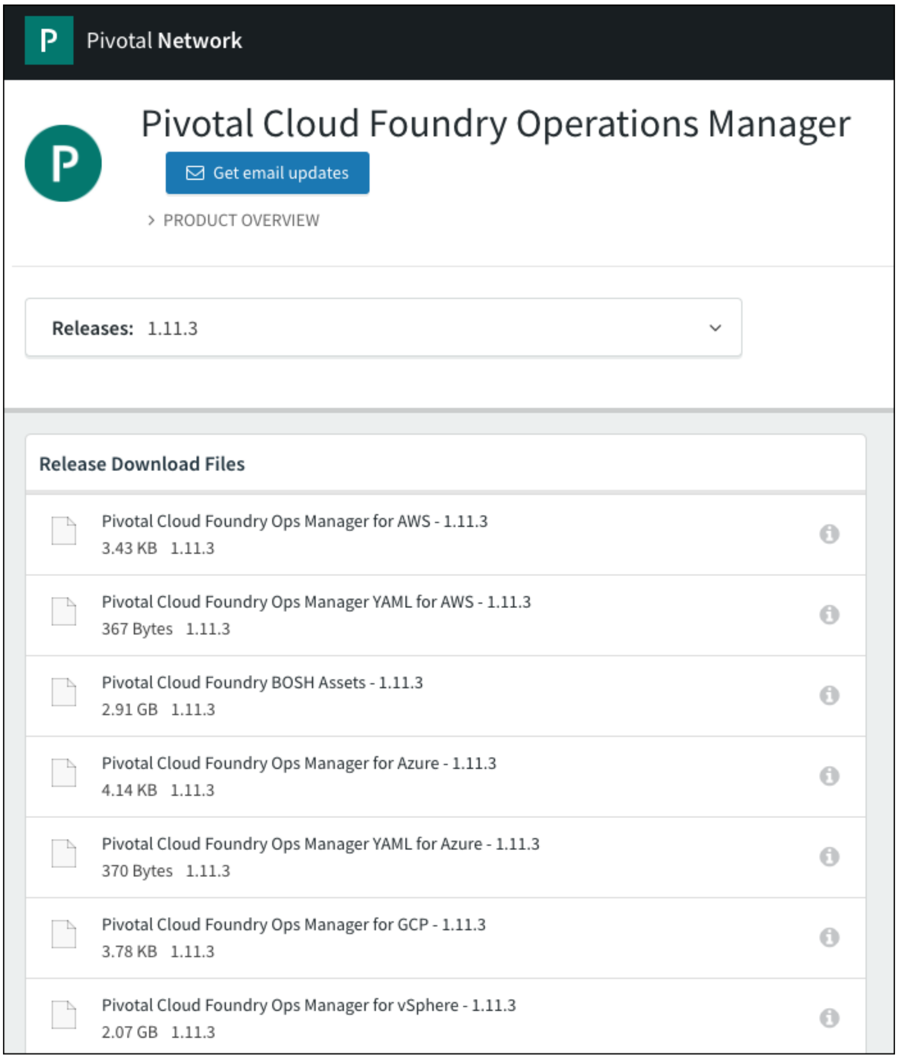
Deploying BOSH and Ops Manager to vSphere

Page last updated:

This topic provides instructions for deploying Ops Manager to VMware vSphere.

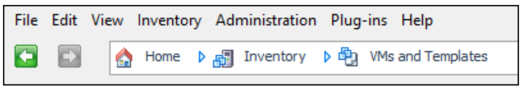
1. Refer to the Known Issues section of the *Ops Manager v2.0 Release Notes* topic before starting.

2. Download the Pivotal Cloud Foundry (PCF) Ops Manager `.ova` file at Pivotal Network. Click the **Pivotal Cloud Foundry** region to access the PCF product page. Use the dropdown menu to select an Ops Manager release.

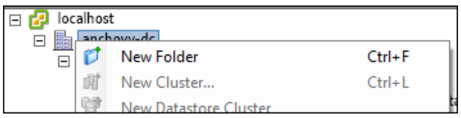


3. Log into vCenter.

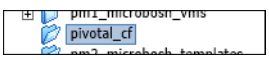
4. Select the **VM and Templates** view.



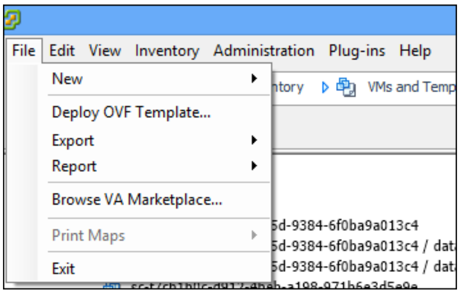
5. Right click on your datacenter and select **New Folder**.



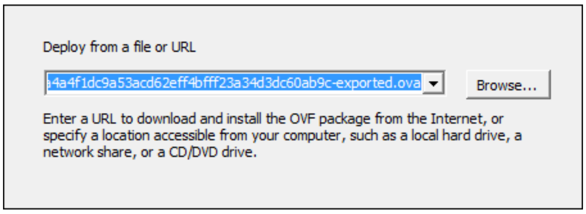
6. Name the folder `pivotal_cf` and select it.



7. Select **File > Deploy OVF Template**.



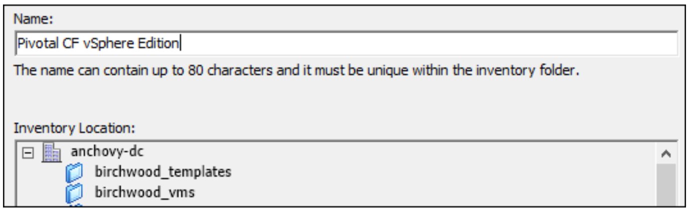
8. Select the .ova file and click **Next**.



9. Review the product details and click **Next**.

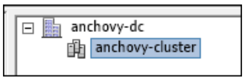
10. Accept the license agreement and click **Next**.

11. Name the virtual machine and click **Next**.



Note: The selected folder is the one you created.

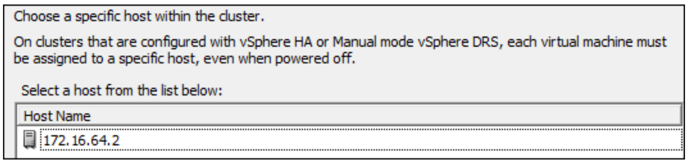
12. Select a vSphere cluster and click **Next**.



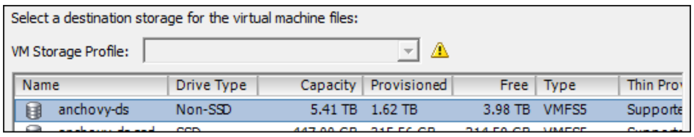
13. If prompted, select a resource pool and click **Next**.

14. If prompted, select a host and click **Next**.

Note: Hardware virtualization must be off if your vSphere host does not support VT-X/EPT. Refer to the Installing Pivotal Cloud Foundry on vSphere topic for more information.

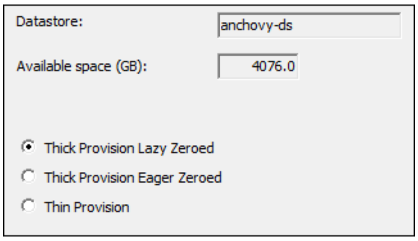


15. Select a storage destination and click **Next**.

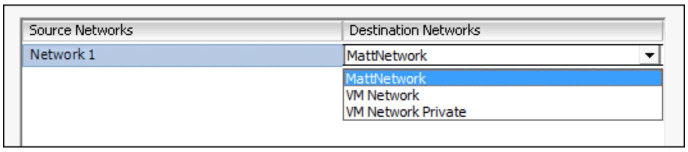


16. Select a disk format and click **Next**. For information about disk formats, see Provisioning a Virtual Disk.

Warning: Ops Manager v2.0 requires a Director VM with at least 8 GB memory.

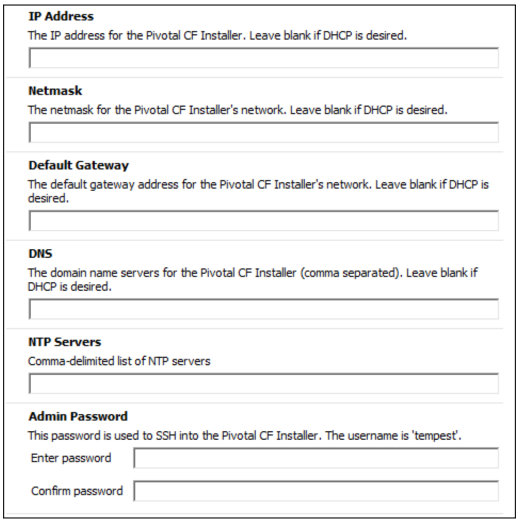


17. Select a network from the drop down list and click **Next**.



18. Enter network information and passwords for the Ops Manager VM admin user and click **Next**.

Note: You must enter a default admin password, or else your Ops Manager VM will not boot up.



Keep this network information. The IP Address will be the location of the Ops Manager interface.

19. Check the **Power on after deployment** checkbox and click **Finish**. Once the VM boots, the interface is available at the IP address you specified.

Note: It is normal to experience a brief delay before the interface is accessible while the web server and VM start up.

20. Create a DNS entry for the IP address that you used for Ops Manager. You must use this fully qualified domain name when you log into Ops Manager in the Installing Pivotal Cloud Foundry on vSphere topic.

Note: Ops Manager security features require you to create a fully qualified domain name to access Ops Manager during the initial configuration.

Return to the Installing Pivotal Cloud Foundry Guide

Configuring Ops Manager on vSphere

Page last updated:

This topic describes how to configure the Ops Manager Director for VMware vSphere.

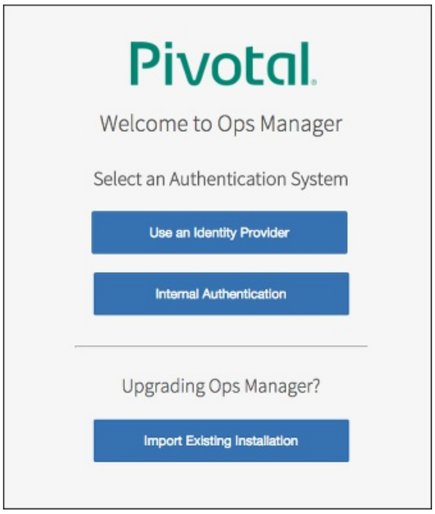
Before you begin this procedure, ensure that you have successfully completed all steps in the Deploying Operations Manager to vSphere topic. After you complete this procedure, follow the instructions in the Configuring Pivotal Application Service (PAS) for vSphere topic.

> **Note**: You can also perform the procedures in this topic using the Ops Manager API. For more information, see the Using the Ops Manager API topic.

Step 1: Set Up Ops Manager

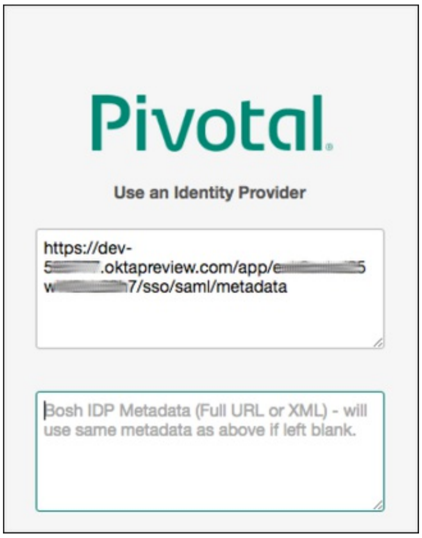
1. Navigate to the fully qualified domain of your Ops Manager in a web browser.

2. The first time you start Ops Manager, you must choose one of the following:
   - **Use an Identity Provider**: If you use an Identity Provider, an external identity server maintains your user database.
   - **Internal Authentication**: If you use Internal Authentication, PCF maintains your user database.



Use an Identity Provider

1. Log in to your IdP console and download the IdP metadata XML. Optionally, if your IdP supports metadata URL, you can copy the metadata URL instead of the XML.



2. Copy the IdP metadata XML or URL to the Ops Manager **Use an Identity Provider** log in page.

> **Note**: The same IdP metadata URL or XML is applied for the BOSH Director. If you use a separate IdP for BOSH, copy the metadata XML or URL from that IdP and enter it into the BOSH IdP Metadata text box in the Ops Manager log in page.

3. Enter your **Decryption passphrase**. Read the **End User License Agreement**, and select the checkbox to accept the terms.

4. Your Ops Manager log in page appears. Enter your username and password. Click **Login**.

5. Download your SAML Service Provider metadata (SAML Relying Party metadata) by navigating to the following URLs:
   - **5a.** Ops Manager SAML service provider metadata: `https://OPS-MAN-FQDN:443/uaa/saml/metadata`
   - **5b.** BOSH Director SAML service provider metadata: `https://BOSH-IP-ADDRESS:8443/saml/metadata`

> **Note**: To retrieve your `BOSH-IP-ADDRESS`, navigate to the **Ops Manager Director** tile > **Status** tab. Record the **Ops Manager Director** IP address.

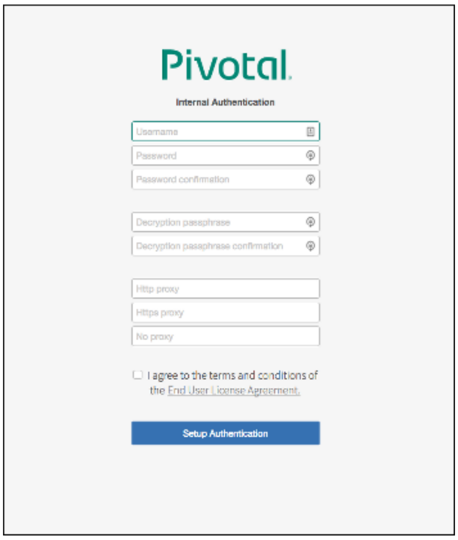
6. Configure your IdP with your SAML Service Provider metadata. Import the Ops Manager SAML provider metadata from Step 5a above to your IdP. If your IdP does not support importing, provide the values below.
   - **Single sign on** URL: `https://OPS-MAN-FQDN:443/uaa/saml/SSO/alias/OPS-MAN-FQDN`
   - **Audience URI** (SP Entity ID): `https://OP-MAN-FQDN:443/uaa`
   - **Name ID**: Email Address
   - SAML authentication requests are always signed

7. Import the BOSH Director SAML provider metadata from Step 5b to your IdP. If the IdP does not support an import, provide the values below.
   - **Single sign on** URL: `https://BOSH-IP:8443/saml/SSO/alias/BOSH-IP`
   - **Audience URI** (SP Entity ID): `https://BOSH-IP:8443`
   - **Name ID**: Email Address
   - SAML authentication requests are always signed

8. Return to the **Ops Manager Director** tile, and continue with the configuration steps below.

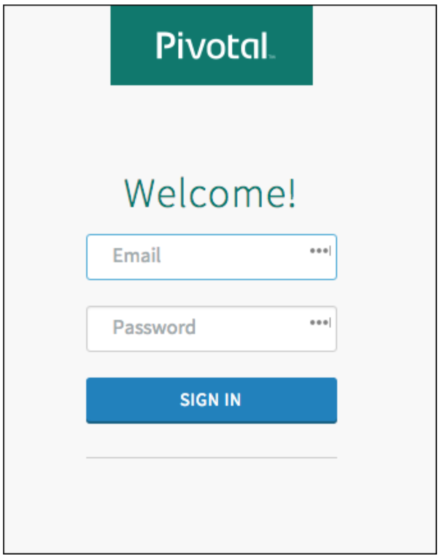
Internal Authentication

1. When redirected to the **Internal Authentication** page, you must complete the following steps:
   - Enter a **Username**, **Password**, and **Password confirmation** to create an Admin user.
   - Enter a **Decryption passphrase** and the **Decryption passphrase confirmation**. This passphrase encrypts the Ops Manager datastore, and is not recoverable.
   - If you are using an **HTTP proxy** or **HTTPS proxy**, follow the instructions in the [Configuring Proxy Settings for the BOSH CPI]() topic.
   - Read the **End User License Agreement**, and select the checkbox to accept the terms.



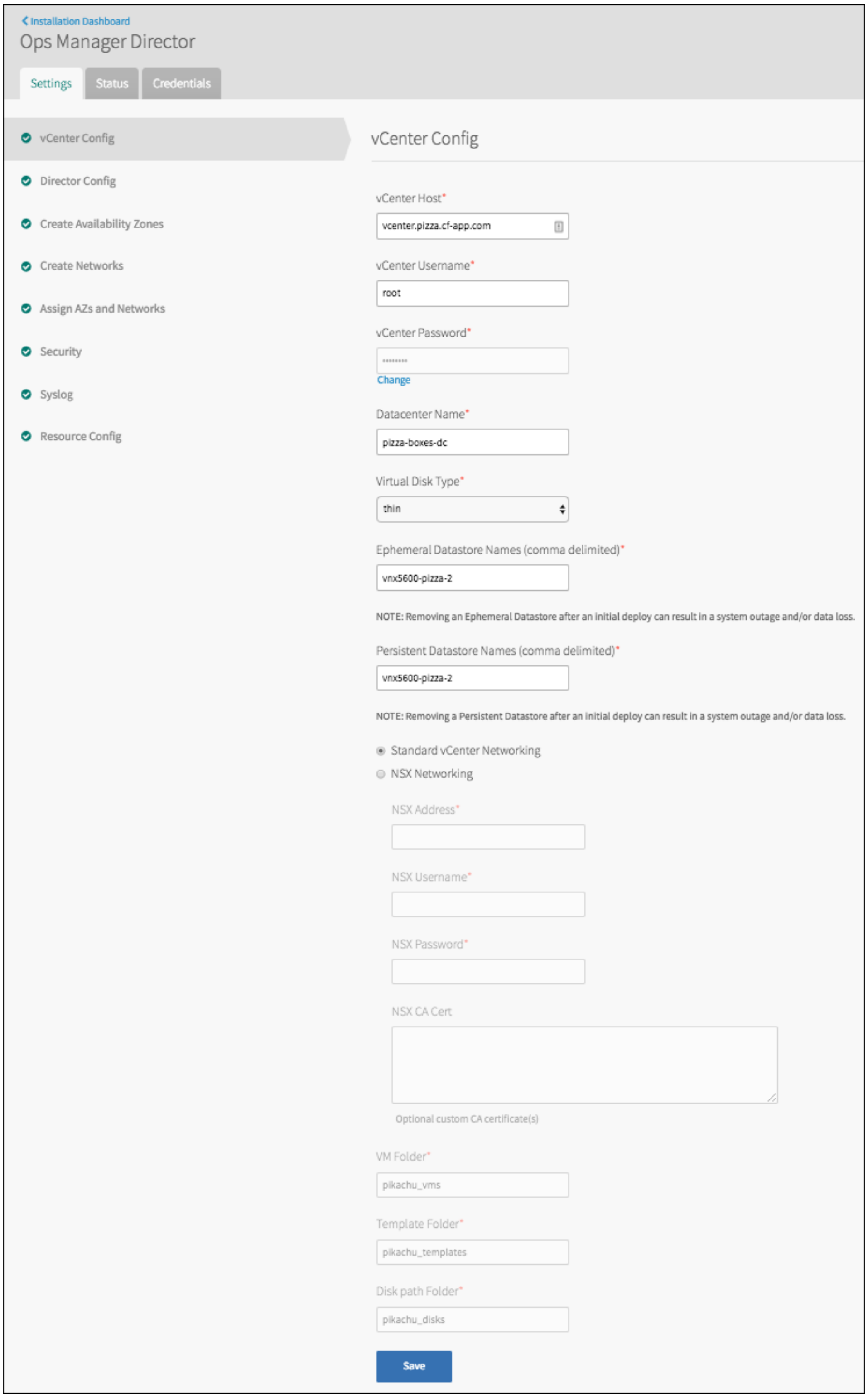
Step 2: vCenter Config Page

1. Log in to Ops Manager with the Admin username and password you created in the previous step.



2. Click the **Ops Manager Director** tile.

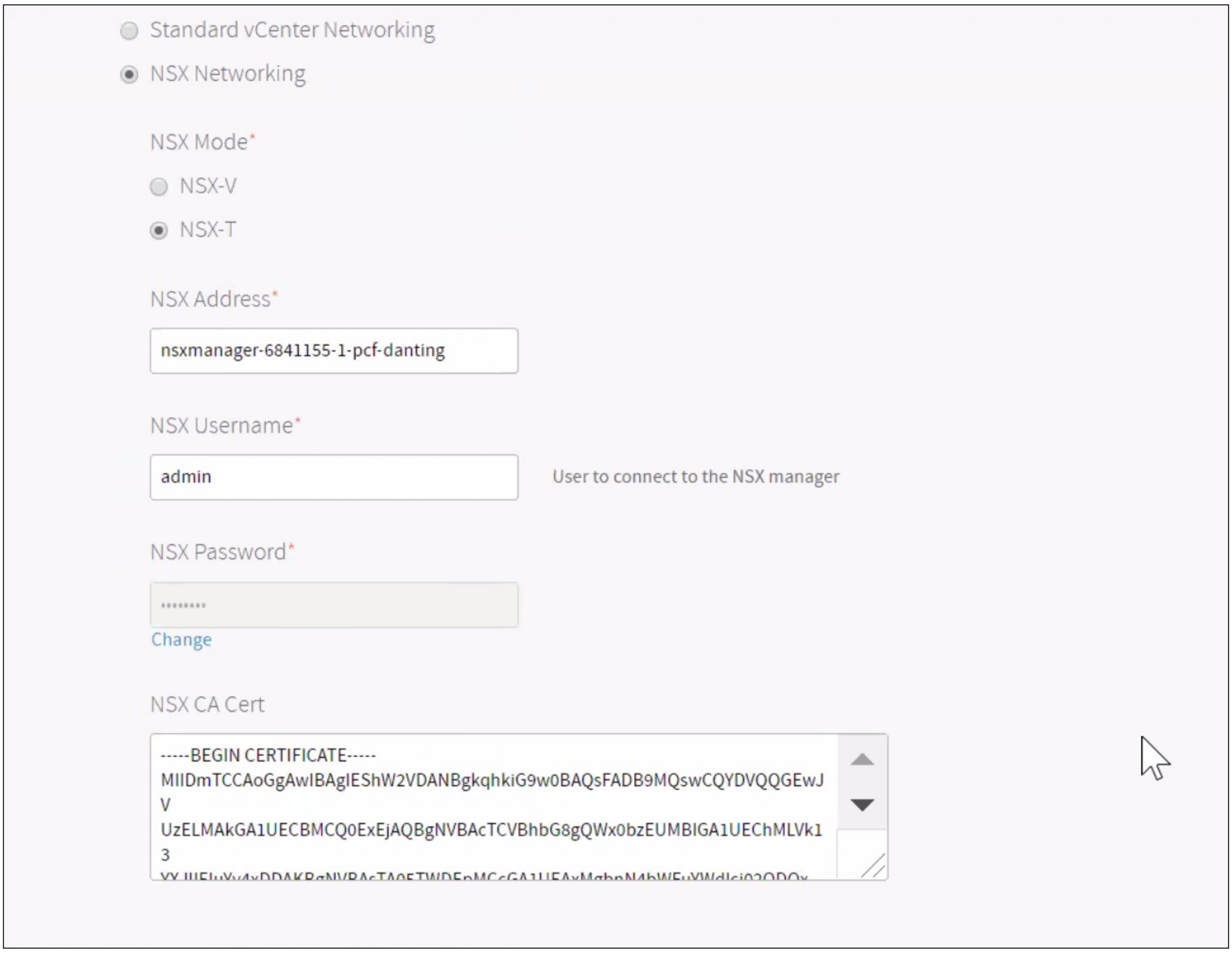
3. Select **vCenter Config**.



4. Enter the following information:
   - **vCenter Host**: The hostname of the vCenter that manages ESXi/vSphere.
   - **vCenter Username**: A vCenter username with create and delete privileges for virtual machines (VMs) and folders.
   - **vCenter Password**: The password for the vCenter user specified above.
   - **Datacenter Name**: The name of the datacenter as it appears in vCenter.
   - **Virtual Disk Type**: The Virtual Disk Type to provision for all VMs. For guidance on the virtual disk type to select, see Provisioning a Virtual Disk in vSphere.
   - **Ephemeral Datastore Names (comma delimited)**: The names of the datastores that store ephemeral VM disks deployed by Ops Manager.
   - **Persistent Datastore Names (comma delimited)**: The names of the datastores that store persistent VM disks deployed by Ops Manager.
   - **VM Folder**: The vSphere datacenter folder (default: `pcf_vms`) where Ops Manager places VMs.
   - **Template Folder**: The vSphere datacenter folder (default: `pcf_templates`) where Ops Manager places VMs.
   - **Disk path Folder**: The vSphere datastore folder (default: `pcf_disk`) where Ops Manager creates attached disk images. You must not nest this

folder.

5. Select a network configuration from one of the following:
   - **Standard vCenter Networking:** This is the default option when upgrading Ops Manager.
   - **NSX Networking:** Select this option to enable VMware NSX Network Virtualization.



6. Configure NSX networking by entering the following information:
   - **NSX Mode**: Select either **NSX-V** or **NSX-T**.
   - **NSX Address**: The address of the NSX manager.
   - **NSX Username**: The username to connect to the NSX manager.
   - **NSX Password**: The password for the username specified above.
   - **NSX CA Cert**: A CA certificate in PEM format that authenticates to the NSX server. Provide the fully qualified domain name if the NSX Manager uses a self-signed SSL certificate.

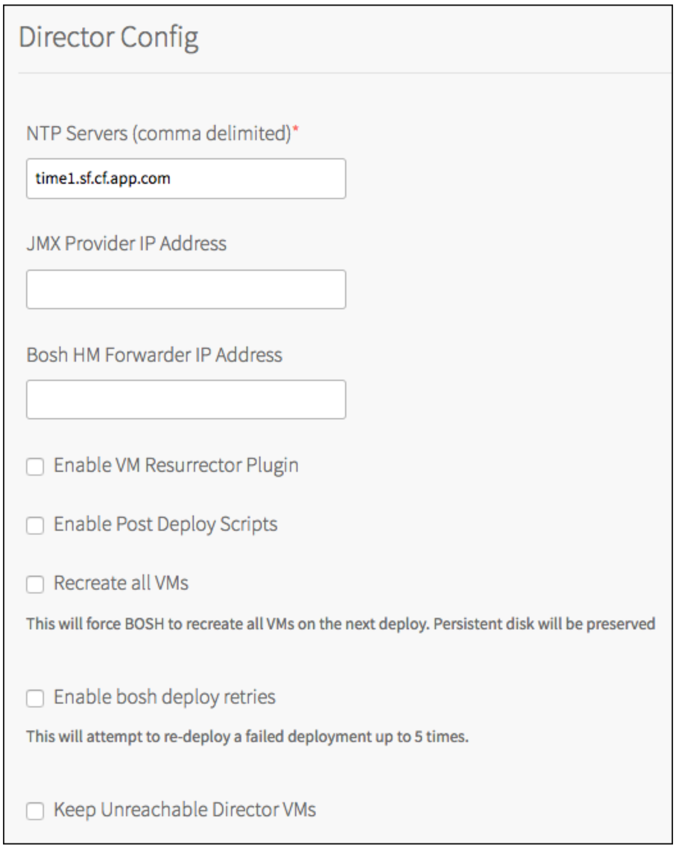
   > **Note**: To update NSX security group and load balancer information, see the Updating NSX Security Group and Load Balancer Information topic.

7. Click **Save**.

> **Note**: After your initial deployment, you will not be able to edit the VM Folder, Template Folder, and Disk path Folder names.

Step 3: Director Config Page

1. Select **Director Config**.



2. In the **NTP Servers (comma delimited)** field, enter your NTP server addresses.

3. Leave the **JMX Provider IP Address** field blank.

> **Note**: Starting from PCF v2.0, BOSH-reported system metrics are available in the Loggregator Firehose by default. Therefore, if you continue to use PCF JMX Bridge for consuming them outside of the Firehose, you may receive duplicate data. To prevent this, leave the **JMX Provider IP Address** field blank. For additional guidance, see BOSH System Metrics Available in Loggregator Firehose.

4. Leave the **Bosh HM Forwarder IP Address** field blank.

> **Note**: Starting from PCF v2.0, BOSH-reported system metrics are available in the Loggregator Firehose by default. Therefore, if you continue to use the BOSH HM Forwarder for consuming them, you may receive duplicate data. To prevent this, leave the **Bosh HM Forwarder IP Address** field blank. For additional guidance, see BOSH System Metrics Available in Loggregator Firehose.

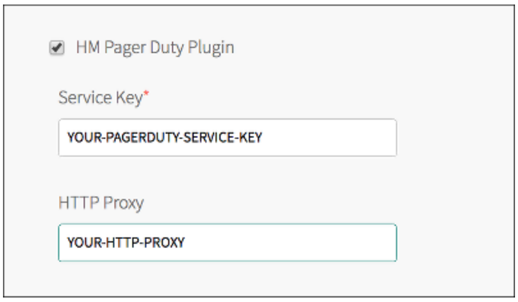
5. Select the **Enable VM Resurrector Plugin** to enable Ops Manager Resurrector functionality and increase PAS availability. For more information, see the Using Ops Manager Resurrector on VMware vSphere topic.

6. Select **Enable Post Deploy Scripts** to run a post-deploy script after deployment. This script allows the job to execute additional commands against a deployment.

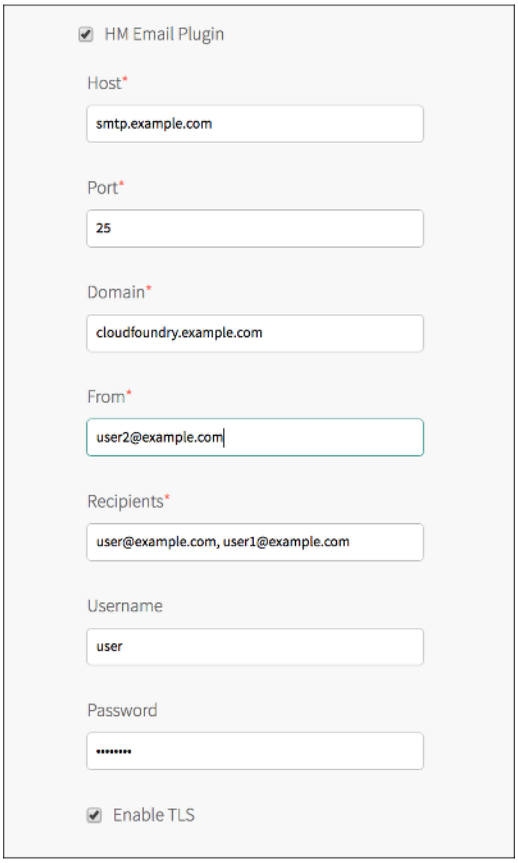
7. Select **Recreate all VMs** to force BOSH to recreate all VMs on the next deploy. This process does not destroy any persistent disk data.

8. Select **Enable bosh deploy retries** if you want Ops Manager to retry failed BOSH operations up to five times.

9. Select **Keep Unreachable Director VMs** if you want to preserve Ops Manager Director VMs after a failed deployment for troubleshooting purposes.

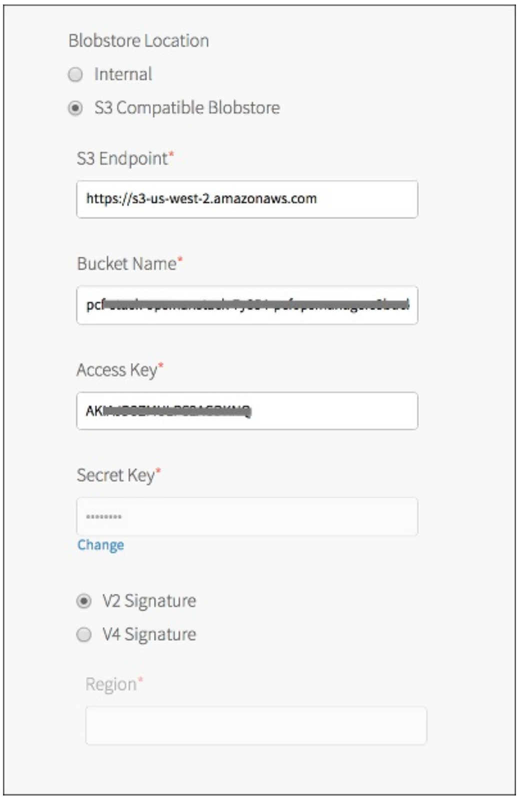


10. Select **HM Pager Duty Plugin** to enable Health Monitor integration with PagerDuty.
    - **Service Key**: Enter your API service key from PagerDuty.
    - **HTTP Proxy**: Enter an HTTP proxy for use with PagerDuty.

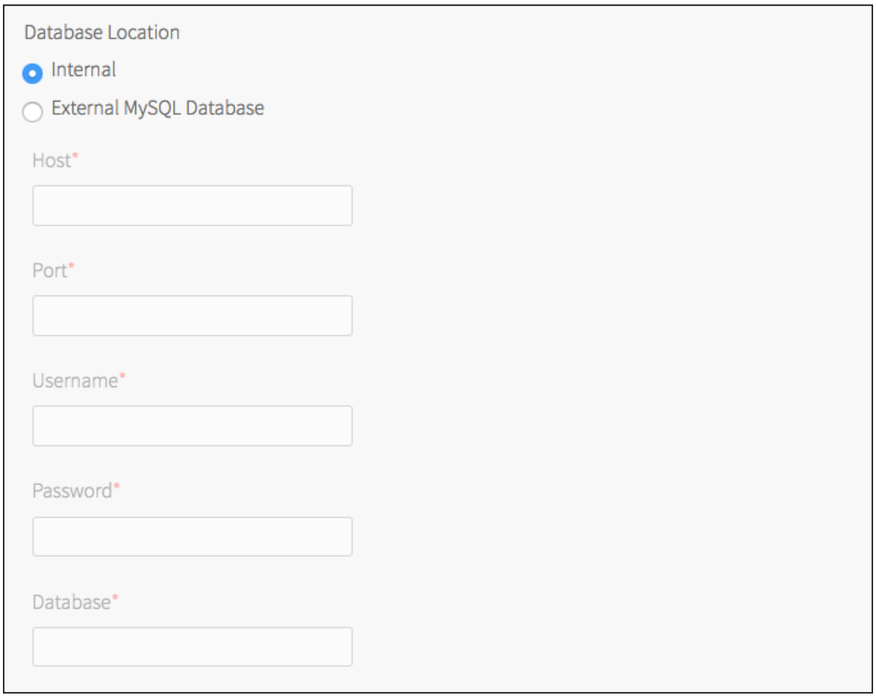


11. Select **HM Email Plugin** to enable Health Monitor integration with email.
    - **Host**: Enter your email hostname.
    - **Port**: Enter your email port number.
    - **Domain**: Enter your domain.
    - **From**: Enter the address for the sender.
    - **Recipients**: Enter comma-separated addresses of intended recipients.
    - **Username**: Enter the username for your email server.
    - **Password**: Enter the password for your email server.
    - **Enable TLS**: Select this checkbox to enable Transport Layer Security.

12. For **Blobstore Location**, Pivotal recommends that you select **Internal**. However, if you select **S3 Compatible Blobstore**, complete the **S3 Endpoint**, **Bucket Name**, **Access Key**, **Secret Key**, **V2 Signature/V4 Signature**, and **Region** with information from your blobstore provider.



13. By default, Pivotal Cloud Foundry (PCF) deploys and manages an **Internal** database for you. If you choose to use an **External MySQL Database**, complete the associated fields with information obtained from your external MySQL Database provider: **Host**, **Port**, **Username**, **Password**, and **Database**.



14. (Optional) **Director Workers** sets the number of workers available to execute Director tasks. This field defaults to `5`.

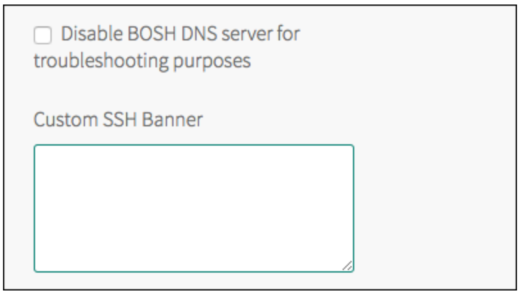
15. (Optional) **Max Threads** sets the maximum number of threads that the Ops Manager Director can run simultaneously. For vSphere, the default value is `32`. Leave the field blank to use this default value. Pivotal recommends that you use the default value unless doing so results in rate limiting or errors on your IaaS.

16. (Optional) To add a custom URL for your Ops Manager Director, enter a valid hostname in **Director Hostname**. You can also use this field to configure a load balancer in front of your Ops Manager Director.

17. (Optional) To disable BOSH DNS, select the **Disable BOSH DNS server for troubleshooting purposes** checkbox. For more information about the BOSH DNS service discovery mechanism, see BOSH DNS Service Discovery for Application Containers in the PAS Release Notes.

    Breaking Change: Do not disable BOSH DNS without consulting Pivotal Support. If you opt out of the BOSH DNS feature, your PCF deployment cannot support NSX-T networking. For more information about disabling BOSH DNS, see Disabling or Opting Out of BOSH DNS in PCF in the Pivotal Knowledge Base.

18. (Optional) To set a custom banner that users see when logging in to the Director using SSH, enter text in the **Custom SSH Banner** field.



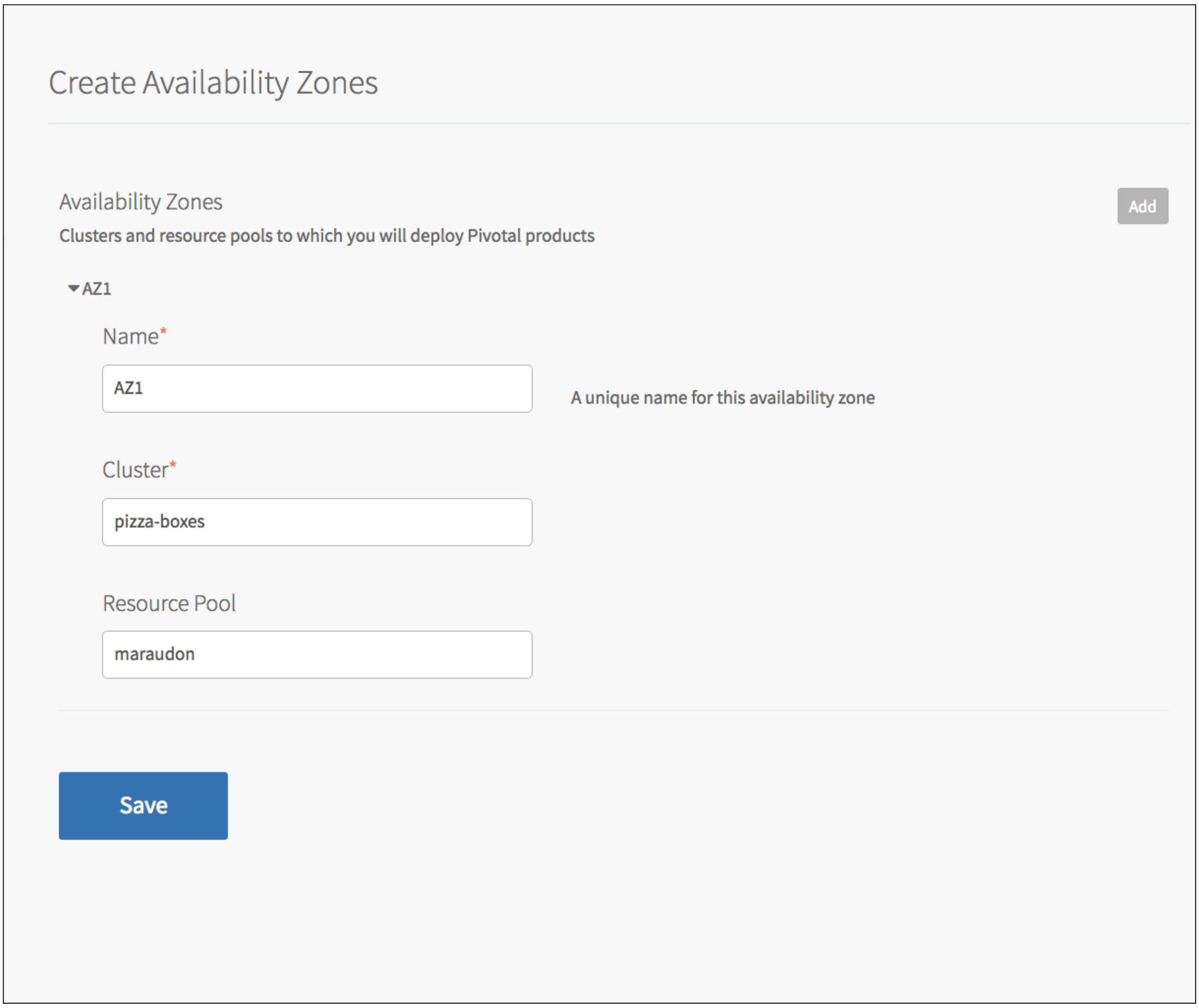
19. Click **Save**.

Note: After your initial deployment, you will not be able to edit the Blobstore and Database locations.

Step 4: Create Availability Zone Page

Ops Manager Availability Zones correspond to your vCenter clusters and resource pools. Multiple Availability Zones allow you to provide high-availability and load balancing to your applications. When you run more than one instance of an application, Ops Manager balances those instances across all of the Availability Zones assigned to the application. At least three availability zones are recommended for a highly available installation of PAS.

1. Select **Create Availability Zones**.



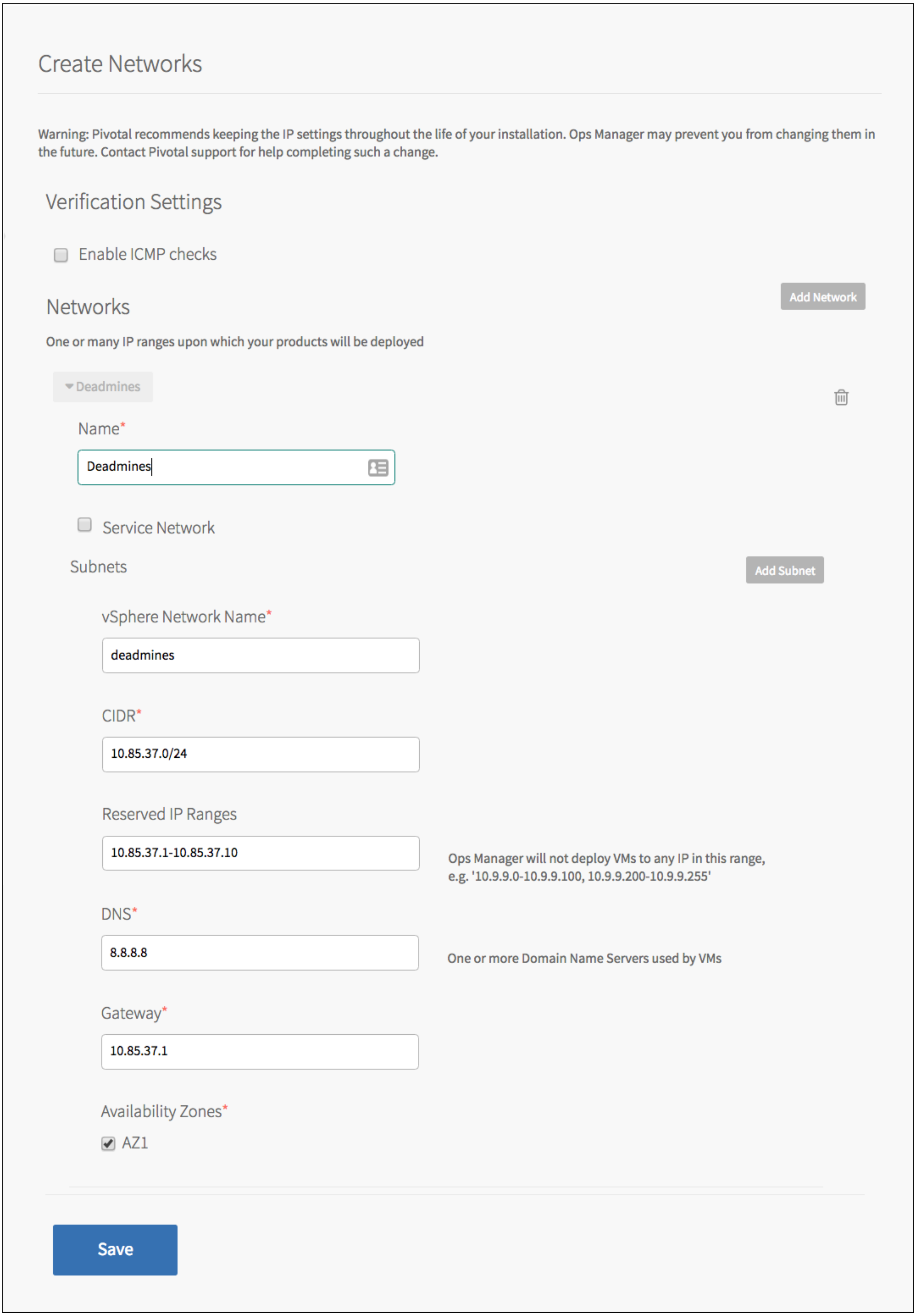
2. Use the following steps to create one or more Availability Zones for your applications to use:
   - Click **Add**.
   - Enter a unique **Name** for the Availability Zone.
   - Enter the name of an existing vCenter **Cluster** to use as an Availability Zone.
   - **(Optional)** Enter the name of a **Resource Pool** in the vCenter cluster that you specified above. The jobs running in this Availability Zone share the CPU and memory resources defined by the pool.

   > **Note**: For more information about using availability zones in vSphere, see the [Understanding Availability Zones in VMware Installations]() topic.

3. Click **Save**.

Step 5: Create Networks Page

1. Select **Create Networks**.



2. Select **Enable ICMP checks** to enable ICMP on your networks. Ops Manager uses ICMP checks to confirm that components within your network are reachable.

3. Use the following steps to create one or more Ops Manager networks:
   - Click **Add Network**.
   - Enter a unique **Name** for the network.
   - If you want to dynamically provision VMs in this network for use with on-demand services, select the **Service Networks** checkbox. When the checkbox is selected, Ops Manager does not provision VMs within the specified CIDR range.
   - Click **Add Subnet** to create one or more subnets for the network.
   - Enter the full path and **vSphere Network Name** as it displays in vCenter. For example, enter `YOUR-DIRECTORY-NAME/YOUR-NETWORK-NAME`. If your vSphere Network Name contains a forward slash character, replace the forward slash with the URL-encoded forward slash character `%2F`.
   - For **CIDR**, enter a valid CIDR block in which to deploy VMs. For example, enter `192.0.2.0/24`.
   - For **Reserved IP Ranges**, enter any IP addresses from the **CIDR** that you want to blacklist from the installation. Ops Manager will not deploy VMs to any address in this range.
   - Enter your **DNS** and **Gateway** IP addresses.
   - Select which **Availability Zones** to use with the network.

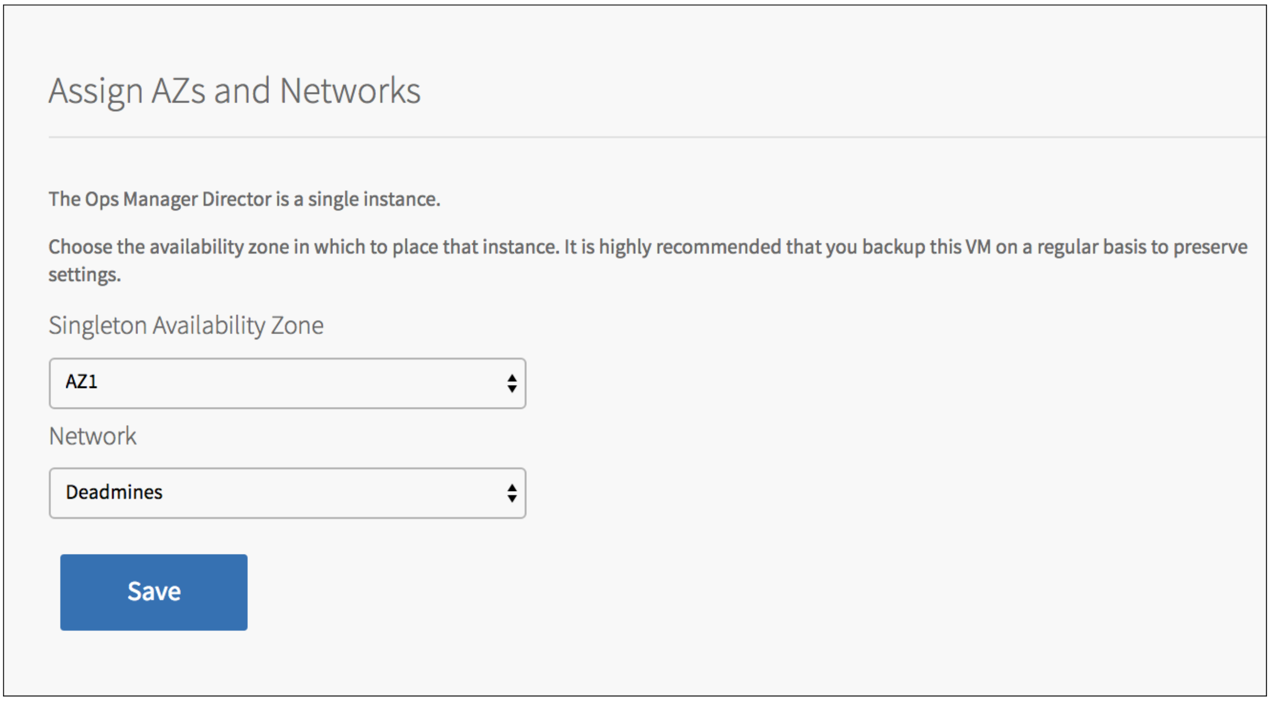
4. Click **Save**.

> **Note**: Multiple networks allow you to place vCenter on a private network and the rest of your deployment on a public network. Isolating vCenter in this manner denies access to it from outside sources and reduces possible security vulnerabilities.

> **Note**: If you are using the Cisco Nexus 1000v Switch, refer to the Using the Cisco Nexus 1000v Switch with Ops Manager topic for more information.

Step 6: Assign AZs and Networks Page

1. Select **Assign AZs and Networks**.



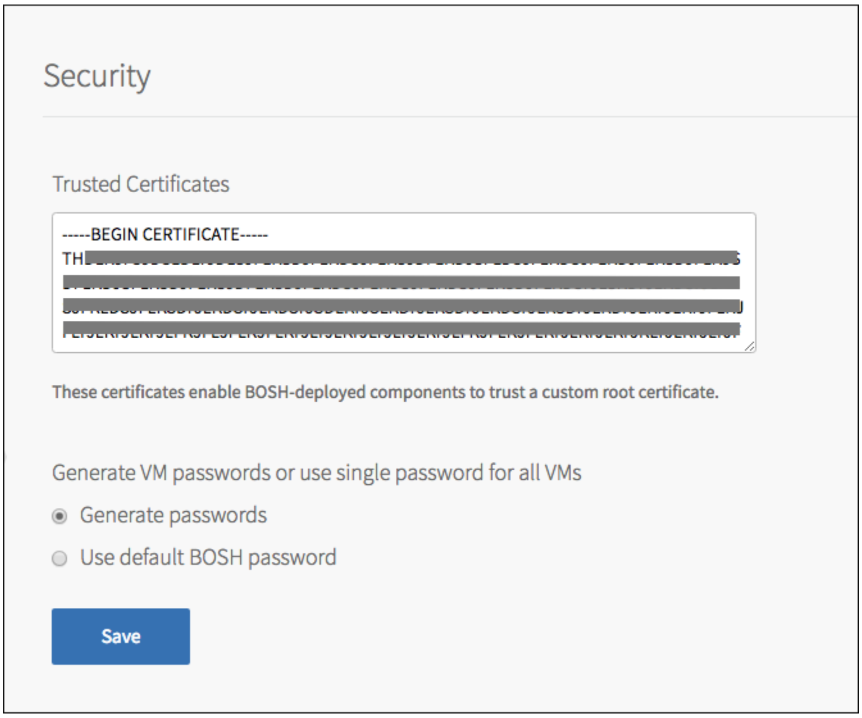
2. Use the drop-down menu to select a **Singleton Availability Zone**. The Ops Manager Director installs in this Availability Zone.

3. Use the drop-down menu to select a **Network** for your Ops Manager Director.

4. Click **Save**.

Step 7: Security Page

1. Select **Security**.



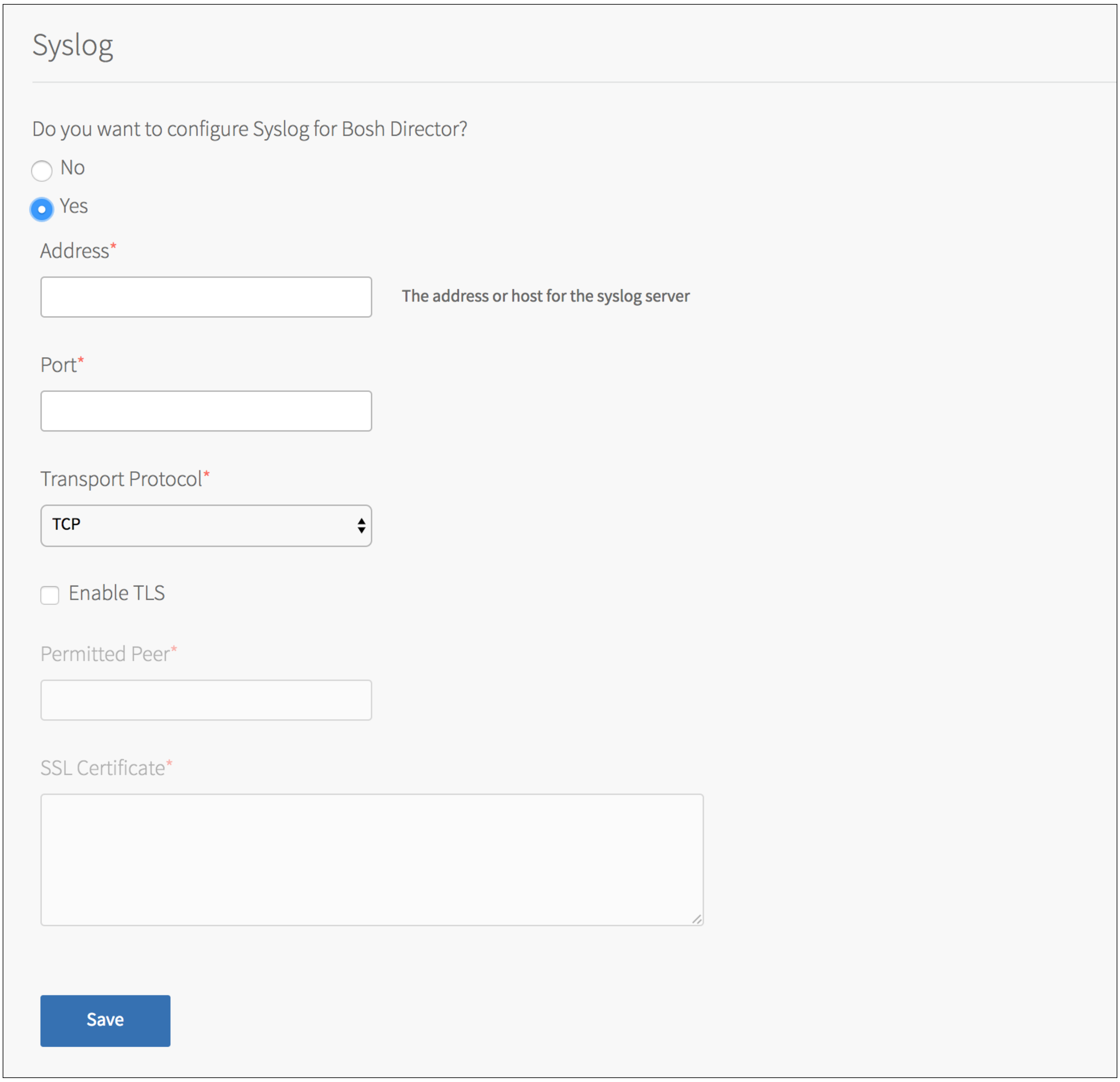
2. In **Trusted Certificates**, enter a custom certificate authority (CA) certificate to insert into your organization's certificate trust chain. This feature enables all BOSH-deployed components in your deployment to trust a custom root certificate. If you want to use Docker Registries for running app instances in Docker containers, use this field to enter your certificate for your private Docker Registry. See the Using Docker Registries topic for more information.

3. Choose **Generate passwords** or **Use default BOSH password**. Pivotal recommends that you use the **Generate passwords** option for greater security.

4. Click **Save**. To view your saved Director password, click the **Credentials** tab.

Step 8: Syslog Page

1. Select **Syslog**.



2. (Optional) To send BOSH Director system logs to a remote server, select **Yes**.

3. In the **Address** field, enter the IP address or DNS name for the remote server.

4. In the **Port** field, enter the port number that the remote server listens on.

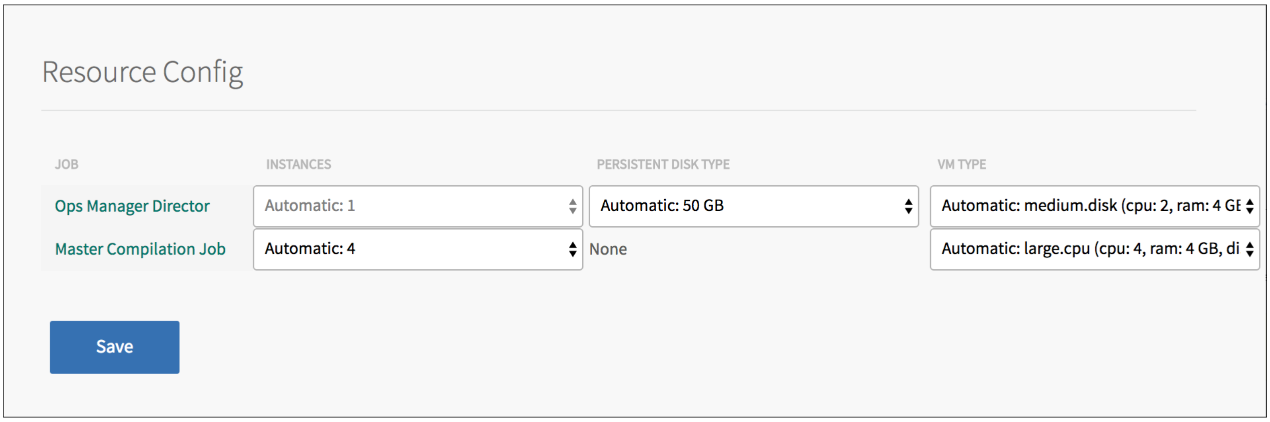
5. In the **Transport Protocol** dropdown menu, select **TCP**, **UDP**, or **RELP**. This selection determines which transport protocol is used to send the logs to the remote server.

6. (Optional) Mark the **Enable TLS** checkbox to use TLS encryption when sending logs to the remote server.
   - In the **Permitted Peer** field, enter either the name or SHA1 fingerprint of the remote peer.
   - In the **SSL Certificate** field, enter the SSL certificate for the remote server.

7. Click **Save**.

Step 9: Resource Config Page

1. Select **Resource Config**.



2. Adjust any values as necessary for your deployment. Under the **Instances**, **Persistent Disk Type**, and **VM Type** fields, choose **Automatic** from the drop-down menu to allocate the recommended resources for the job. If the **Persistent Disk Type** field reads **None**, the job does not require persistent disk space.

   > **Note**: Ops Manager v1.11 requires a Director VM with at least 8 GB memory.

   > **Note**: If you set a field to **Automatic** and the recommended resource allocation changes in a future version, Ops Manager automatically uses the updated recommended allocation.

3. Click **Save**.

Step 10: Complete the Ops Manager Director Installation

1. Click the **Installation Dashboard** link to return to the Installation Dashboard.
2. Click **Apply Changes** on the right navigation.
3. After you complete this procedure, follow the instructions in the Configuring PAS for vSphere topic.

Configuring PAS for vSphere

Page last updated:

This topic describes how to configure the Pivotal Application Service (PAS) components that you need to run Pivotal Cloud Foundry (PCF) for VMware vSphere.

Note: If you plan to install the IPsec add-on, you must do so before installing any other tiles. Pivotal recommends installing IPsec immediately after Ops Manager, and before installing the PAS tile.

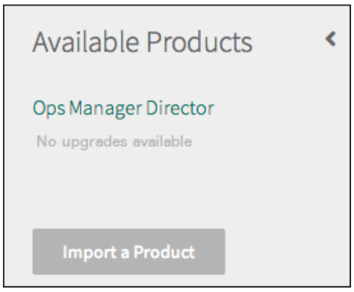
Step 1: Add PAS to Ops Manager

1. If you have not already downloaded PAS, log in to Pivotal Network, and click on PAS.

2. From the **Releases** drop-down, select the release to install and choose one of the following:

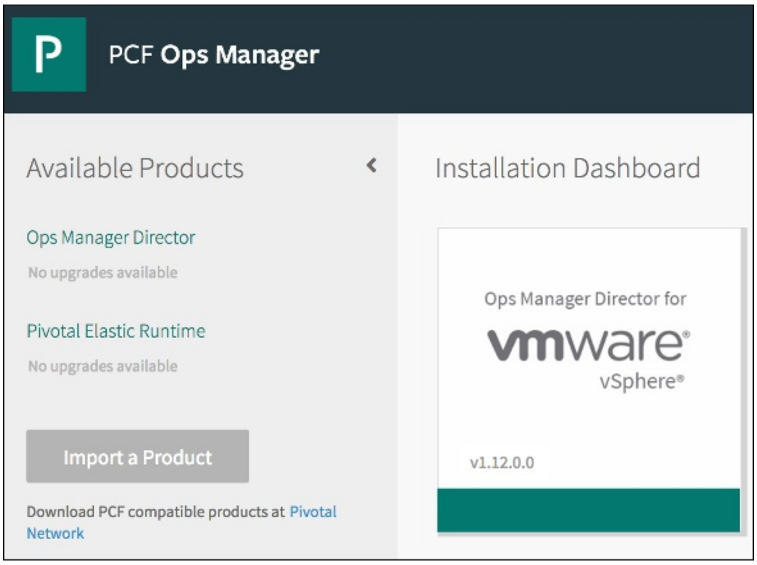
    a. Click PAS to download the PAS `.pivotal` file
    b. Click PCF Small Footprint PAS to download the Small Footprint Runtime `.pivotal` file. For more information, see Getting Started with Small Footprint Runtime.

3. From the Available Products view, click **Import a Product**.



4. Select the PAS `.pivotal` file that you downloaded from Pivotal Network and click **Open**. After the import completes, PAS appears in the Available Products view.

5. In the Available Products view, hover over PAS and click **Add**.



6. Click the PAS tile in the Installation Dashboard.

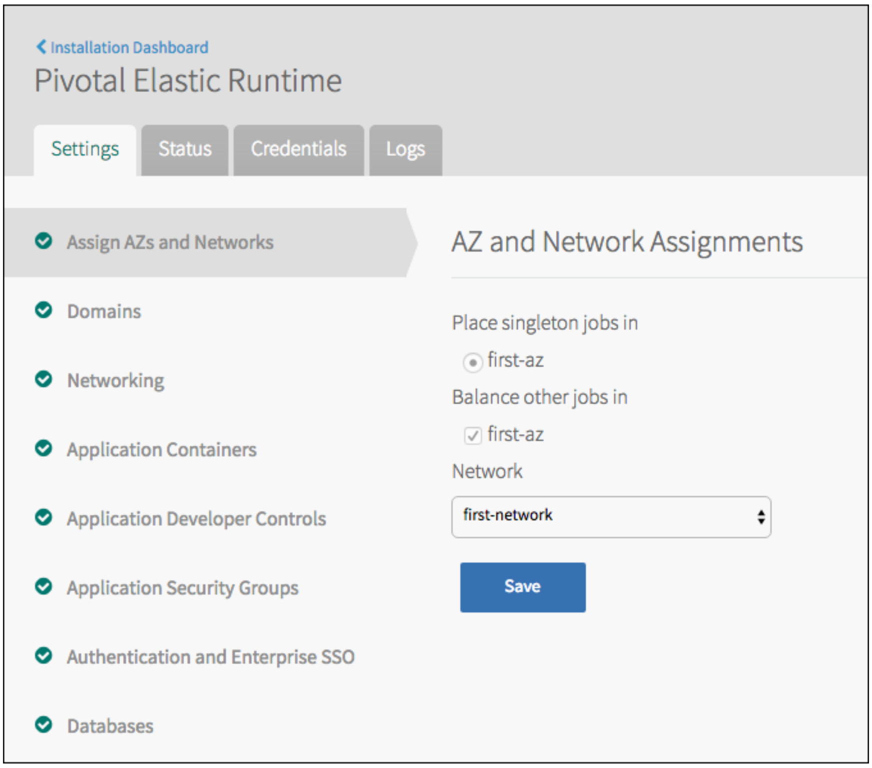
Step 2: Assign Availability Zones and Networks

1. Select **Assign AZs and Networks**. These are the Availability Zones that you create when configuring Ops Manager Director.

2. **(vSphere Only)** Select an Availability Zone under **Place singleton jobs**. Ops Manager runs any job with a single instance in this Availability Zone.

3. **(vSphere Only)** Select one or more Availability Zones under **Balance other jobs**. Ops Manager balances instances of jobs with more than one instance across the Availability Zones that you specify.

4. From the **Network** drop-down box, choose the network on which you want to run PAS.

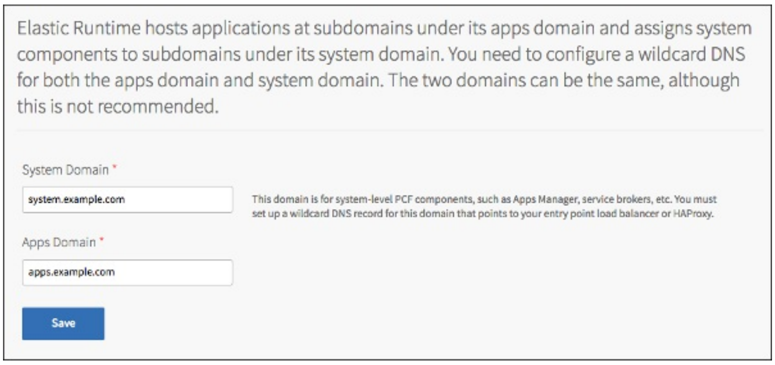


5. Click **Save**.

> **Note**: When you save this form, a verification error displays because the PCF security group blocks ICMP. You can ignore this error.

Step 3: Configure Domains

1. Select **Domains**.



2. Enter the system and application domains.
   - The **System Domain** defines your target when you push apps to PAS.
   - The **Apps Domain** defines where PAS should serve your apps.

> **Note:** Pivotal recommends that you use the same domain name but different subdomain names for your system and app domains. Doing so allows you to use a single wildcard certificate for the domain while preventing apps from creating routes that overlap with system routes. For example, name your system domain `system.EXAMPLE.com` and your apps domain `apps.EXAMPLE.com`.

3. Click **Save**.

Step 4: Configure Networking

1. Select **Networking**.

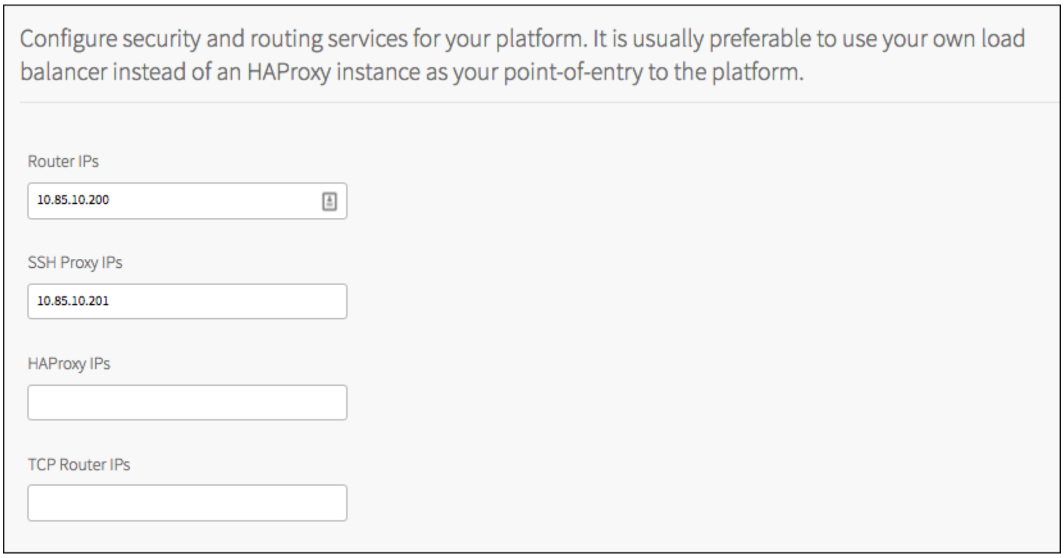
2. The values you enter in the **Router IPs** and **HAProxy IPs** fields depend on whether you are using HAProxy in your deployment. Use the table below to determine how to complete these fields.

> **Note:** If you choose to assign specific IP addresses in either the **Router IPs** or **HAProxy IPs** field, ensure that these IP addresses are in the subnet that you configured for PAS in Ops Manager.

Using HAProxy?	Router IPs Field	HAProxy IPs Field
No	1. Choose IP addresses from the subnet you configured in Ops Manager. 2. Enter these IP addresses in the **Router IPs** field. You should specify more than one IP address for high availability. 3. Configure your load balancer to forward requests for the domains that you have configured for your deployment to these IP addresses.	Leave this field blank.
Yes	Leave this field blank.	1. Choose IP addresses from the subnet you configured in Ops Manager. 2. Enter these IP addresses in the **HAProxy IPs** field. You should specify more than one IP address for high availability. 3. Configure your load balancer to forward requests for the domains you have configured for your deployment to these IP addresses.

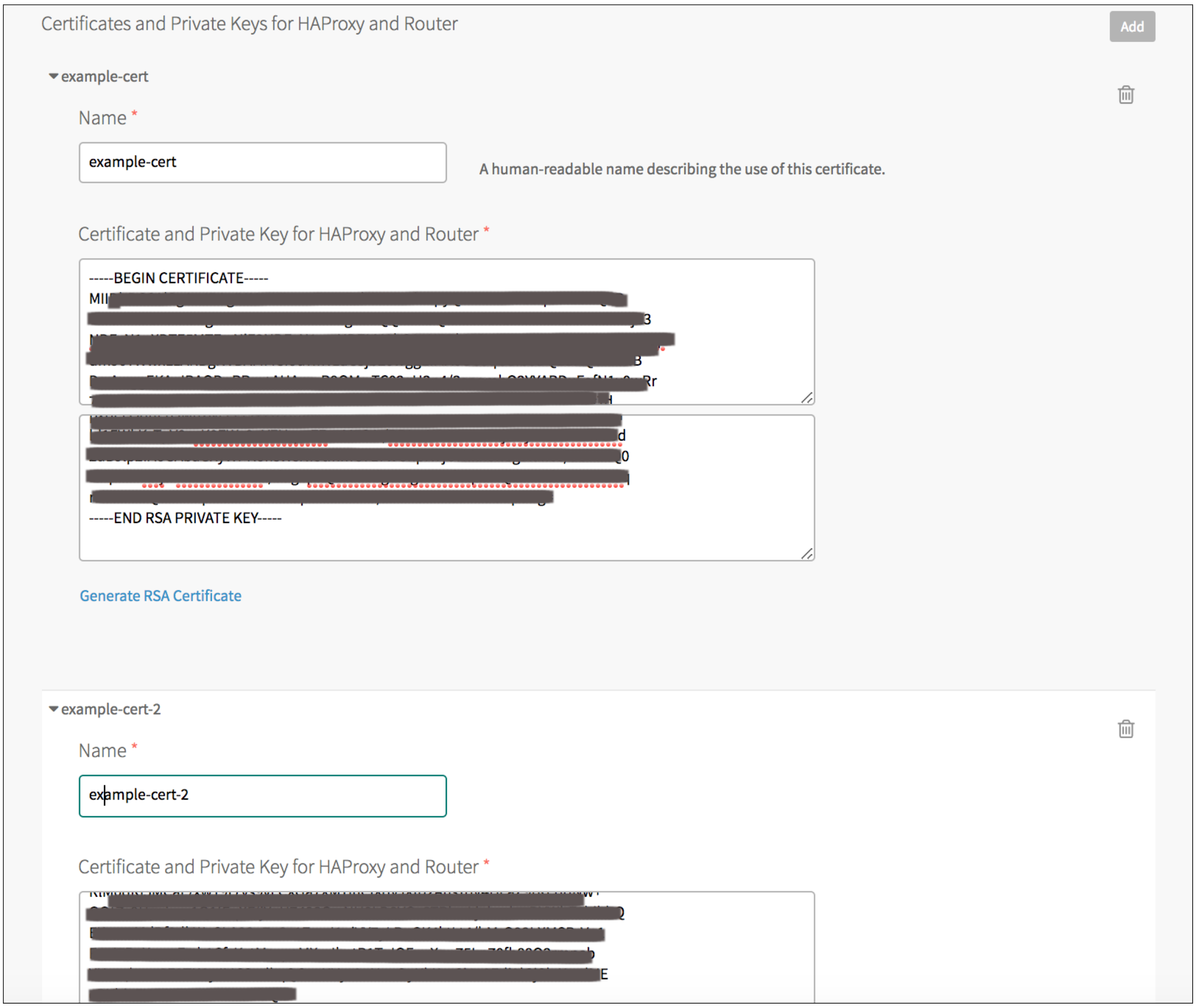
3. (Optional) In **SSH Proxy IPs**, add the IP address for your Diego Brain, which will accept requests to SSH into application containers on port `2222`.

4. (Optional) In **TCP Router IPs**, add the IP address(es) you would like assigned to the TCP Routers. You enable this feature at the bottom of this screen.



5. Under **Certificates and Private Key for HAProxy and Router**, you must provide at least one **Certificate and Private Key** name and certificate keypair for HAProxy and Gorouter. The HAProxy and Gorouter are enabled to receive TLS communication by default. You can configure multiple certificates for HAProxy and Gorouter.

   a. Click the **Add** button to add a name for the certificate chain and its private keypair. This certificate is the default used by Gorouter and HAProxy.

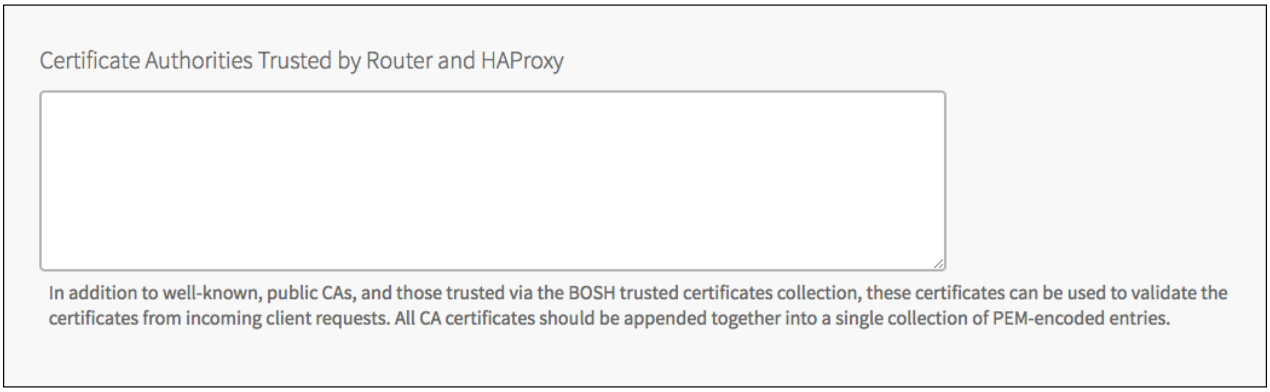
   

   You can either provide a certificate signed by a Certificate Authority (CA) or click on the **Generate RSA Certificate** link to generate a self-signed certificate in Ops Manager.

   b. If you want to configure multiple certificates for HAProxy and Gorouter, click the **Add** button and fill in the appropriate fields for each additional certificate keypair.

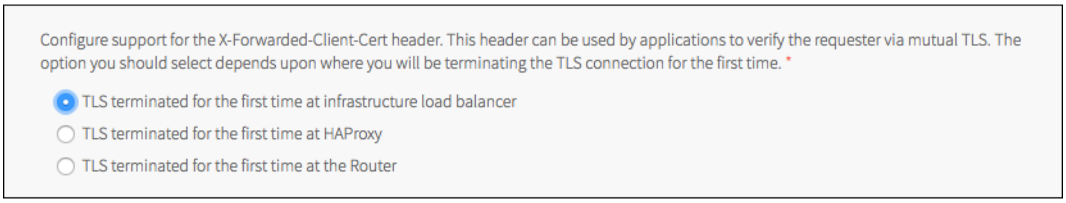
For details about generating certificates in Ops Manager for your wildcard system domains, see the [Providing a Certificate for Your SSL/TLS Termination Point] topic.

6. (Optional) When validating client requests using mutual TLS, the Gorouter trusts multiple certificate authorities (CAs) by default. If you want to configure the Gorouter and HAProxy to trust additional CAs, enter your CA certificates under **Certificate Authorities Trusted by Router and HAProxy**. All CA certificates should be appended together into a single collection of PEM-encoded entries.



7. In the **Minimum version of TLS supported by HAProxy and Router** field, select the minimum version of TLS to use in HAProxy and Router communications. HAProxy and Router use TLS v1.2 by default. If you need to accommodate clients that use an older version of TLS, select a lower minimum version. For a list of TLS ciphers supported by the Gorouter, see [Securing Traffic into Cloud Foundry].

8. Under **Configure support for the X-Forwarded-Client-Cert header**, configure PCF handles `x-forwarded-client-cert` (XFCC) HTTP headers based on where TLS is terminated for the first time in your deployment.



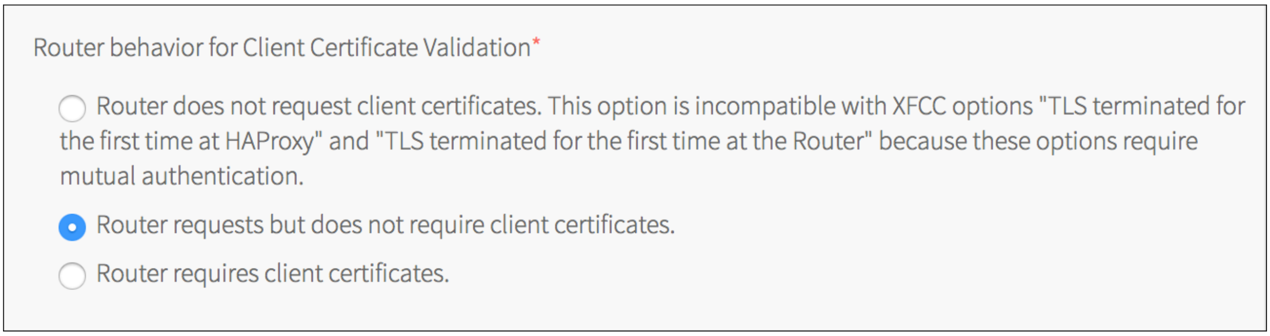
The following table indicates which option to choose based on your deployment layout.

If your deployment is configured as follows:	Then select the following option:	Additional notes:
◦ The Load Balancer is terminating TLS, and ◦ Load balancer is configured to put the client certificate from a mutual authentication TLS handshake into the X-Forwarded-Client-Cert HTTP header	TLS terminated for the first time at infrastructure load balancer (default).	Both HAProxy and the Gorouter forward the XFCC header when included in the request.
◦ The Load Balancer is configured to pass through the TLS handshake via TCP to the instances of HAProxy, and ◦ HAProxy instance count is > 0	TLS terminated for the first time at HAProxy.	HAProxy sets the XFCC header with the client certificate received in the TLS handshake. The Gorouter forwards the header. **Breaking Change**: In the **Router behavior for Client Certificates** field, you cannot select the **Router does not request client certificates** option.
		The Gorouter strips the XFCC header if it is included in the request and forwards the client certificate received in the TLS handshake in a new

◦ The Load Balancer is configured to pass through the TLS handshake via TCP to instances of the Gorouter	TLS terminated for the first time at the Gorouter.	XFCC header. If you have deployed instances of HAProxy, app traffic bypasses those instances in this configuration. If you have also configured your load balancer to route requests for ssh directly to the Diego Brain, consider reducing HAProxy instances to 0. **Breaking Change**: In the **Router behavior for Client Certificates** field, you cannot select the **Router does not request client certificates** option.

For a description of the behavior of each configuration option, see [Forward Client Certificate to Applications](#).

9. To configure Gorouter behavior for handling client certificates, select one of the following options in the **Router behavior for Client Certificate Validation** field.



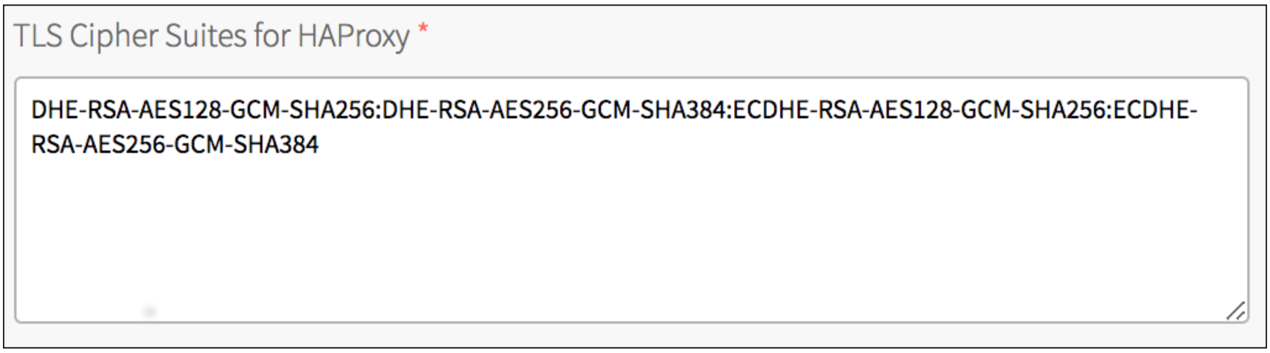
   - **Router does not request client certificates.** This option is incompatible with the XFCC configuration options **TLS terminated for the first time at HAProxy** and **TLS terminated for the first time at the Router** in PAS because these options require mutual authentication. As client certificates are not requested, client will not provide them, and thus validation of client certificates will not occur.
   - **Router requests but does not require client certificates.** Gorouter requests client certificates in TLS handshakes, validates them when presented, but does not require them. This is the default configuration.
   - **Router requires client certificates.** Gorouter validates that the client certificate is signed by a Certificate Authority that Gorouter trusts. If Gorouter cannot validate the client certificate, the TLS handshake fails.

10. In the **TLS Cipher Suites for Router** field, specify the TLS cipher suites to use for TLS handshakes between the Gorouter and downstream clients like load balancers or HAProxy. Use an ordered, colon-delimited list of Golang-supported TLS cipher suites in the OpenSSL format. The recommended setting is `ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384`. Operators should verify that the ciphers are supported by any clients or downstream components that will initiate TLS handshakes with the Gorouter. For a list of TLS ciphers supported by the Gorouter, see [Securing Traffic into Cloud Foundry](#).

Verify that whatever client is participating in the TLS handshake with the Gorouter has at least one cipher suite in common with the Gorouter.

> **Note**: Specify cipher suites that are supported by the versions configured in the **Minimum version of TLS supported by HAProxy and Router** field.

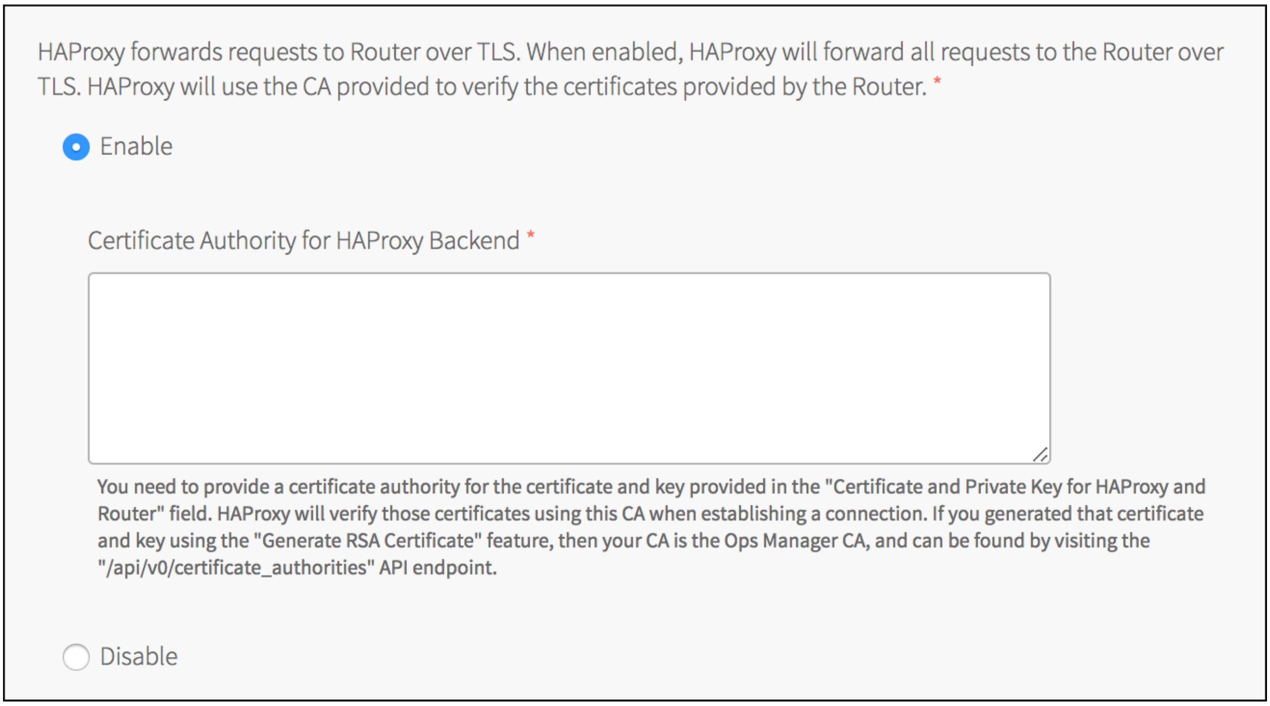
11. In the **TLS Cipher Suites for HAProxy** field, specify the TLS cipher suites to use in HAProxy for TLS handshakes between HAProxy and its clients such as load balancers and Gorouter. Use an ordered, colon-delimited list of TLS cipher suites in the OpenSSL format. The recommended setting: `DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384`
Operators should verify that the ciphers are supported by any clients or downstream components that will initiate TLS handshakes with the HAProxy.



Verify that whatever clients are participating in the TLS handshake with HAProxy have at least one cipher suite in common with HAProxy.

> **Note**: Specify cipher suites that are supported by the versions configured in the **Minimum version of TLS supported by HAProxy and Router** field.

12. Under **HAProxy forwards requests to Router over TLS**, select **Enable** or **Disable** based on your deployment layout.



- **Enable HAProxy forwarding of requests to Router over TLS**

If you want to:	Encrypt communication between HAProxy and the Gorouter
Then configure the following:	1. Leave **Enable** selected. 2. In the **Certificate Authority for HAProxy Backend** field, specify the Certificate Authority (CA) that signed the certificate you configured in the **Certificate and Private Key for HAProxy and Router** field. **Note**: If you used the **Generate RSA Certificate** link to generate a self-signed certificate, then the CA to specify is the Ops Manager CA, which you can locate at the CA endpoint in the Ops Manager API. 3. Make sure that Gorouter and HAProxy have TLS cipher suites in common in the **TLS Cipher Suites for Router** and **TLS Cipher Suites for HAProxy** fields.
See also:	◦ Terminating SSL/TLS at the Load Balancer and Gorouter ◦ Providing a Certificate for Your SSL/TLS Termination Point ◦ Using the Ops Manager API

- **Disable HAProxy forwarding of requests to Router over TLS**

If you want to:	Use non-encrypted communication between HAProxy and Gorouter, or you are not using HAProxy

Then configure the following:	1. Select **Disable**. 2. If you are not using HAProxy, set the number of HAProxy job instances to `0` on the **Resource Config** page. See Disable Unused Resources.
See also:	◦ Terminating SSL/TLS at the Gorouter Only ◦ Terminating SSL/TLS at the Load Balancer Only

13. If you are not using SSL encryption or if you are using self-signed certificates, select **Disable SSL certificate verification for this environment**. Selecting this checkbox also disables SSL verification for route services.

    Note: For production deployments, Pivotal does not recommend disabling SSL certificate verification.

14. (Optional) If you want HAProxy or the Gorouter to reject any HTTP (non-encrypted) traffic, select the **Disable HTTP on HAProxy and Gorouter** checkbox. When selected, HAProxy and Gorouter will not listen on port 80.

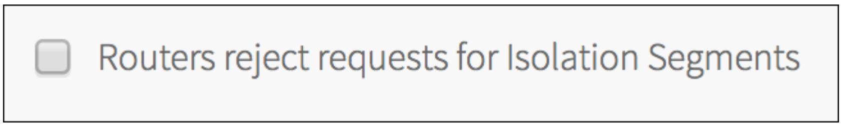
    

15. (Optional) Select the **Disable insecure cookies on the Router** checkbox to set the secure flag for cookies generated by the router.

16. (Optional) To disable the addition of Zipkin tracing headers on the Gorouter, deselect the **Enable Zipkin tracing headers on the router** checkbox. Zipkin tracing headers are enabled by default. For more information about using Zipkin trace logging headers, see Zipkin Tracing in HTTP Headers.

17. (Optional) To stop the Router from writing access logs to local disk, deselect the **Enable Router to write access logs locally** checkbox. You should consider disabling this checkbox for high traffic deployments since logs may not be rotated fast enough and can fill up the disk.

18. By default, the PAS routers handle traffic for applications deployed to an isolation segment created by the PCF Isolation Segment tile. To configure the PAS routers to reject requests for applications within isolation segments, select the **Routers reject requests for Isolation Segments** checkbox.

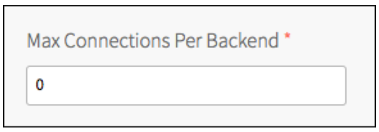
    Do not enable this option without deploying routers for each isolation segment. See the following topics for more information:
    - Installing PCF Isolation Segment
    - Sharding Routers for Isolation Segments

19. In the **Choose whether or not to enable route services** section, choose either **Enable route services** or **Disable route services**. Route services are a class of marketplace services that perform filtering or content transformation on application requests and responses. See the Route Services topic for details.

20. (Optional) If you want to limit the number of app connections to the backend, enter a value in the **Max Connections Per Backend** field. You can use this field to prevent a poorly behaving app from all the connections and impacting other apps.

    To choose a value for this field, review the peak concurrent connections received by instances of the most popular apps in your deployment. You can determine the number of concurrent connections for an app from the `httpStartStop` event metrics emitted for each app request.

    If your deployment uses PCF Metrics, you can also obtain this peak concurrent connection information from Network Metrics. The default value is

    `500`.

21. Enter a value for **Router Max Idle Keepalive Connections**. See Considerations for Configuring max_idle_connections.

22. (Optional) To accommodate larger uploads over connections with high latency, increase the number of seconds in the **Router Timeout to Backends** field.

23. (Optional) Use the **Frontend Idle Timeout for Gorouter and HAProxy** field to help prevent connections from your load balancer to Gorouter or

HAProxy from being closed prematurely. The value you enter sets the duration, in seconds, that Gorouter or HAProxy maintains an idle open connection from a load balancer that supports keep-alive.

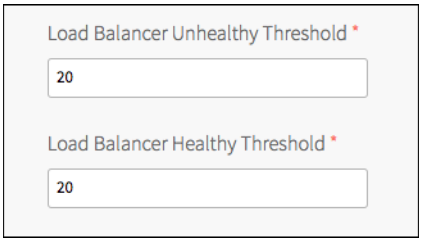
In general, set the value higher than your load balancer's backend idle timeout to avoid the race condition where the load balancer sends a request before it discovers that Gorouter or HAProxy has closed the connection.

See the following table for specific guidance and exceptions to this rule:

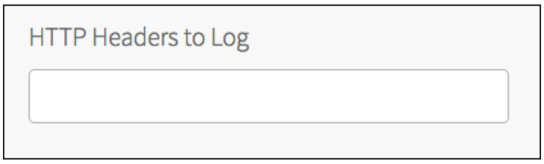
IaaS	Guidance
AWS	AWS ELB has a default timeout of 60 seconds, so Pivotal recommends a value greater than `60`.
Azure	By default, Azure load balancer times out at 240 seconds without sending a TCP RST to clients, so as an exception, Pivotal recommends a value lower than `240` to force the load balancer to send the TCP RST.
GCP	GCP has a default timeout of 600 seconds, so Pivotal recommends a value greater than `600`.
Other	Set the timeout value to be greater than that of the load balancer's backend idle timeout.

24. (Optional) Increase the value of **Load Balancer Unhealthy Threshold** to specify the amount of time, in seconds, that the router continues to accept connections before shutting down. During this period, healthchecks may report the router as unhealthy, which causes load balancers to failover to other routers. Set this value to an amount greater than or equal to the maximum time it takes your load balancer to consider a router instance unhealthy, given contiguous failed healthchecks.

25. (Optional) Modify the value of **Load Balancer Healthy Threshold**. This field specifies the amount of time, in seconds, to wait until declaring the Router instance started. This allows an external load balancer time to register the Router instance as healthy.

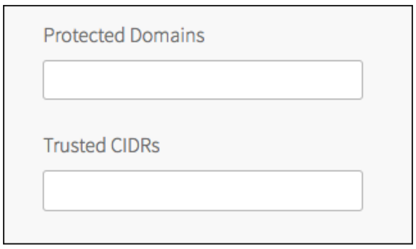


26. (Optional) If app developers in your organization want certain HTTP headers to appear in their app logs with information from the Gorouter, specify them in the **HTTP Headers to Log** field. For example, to support app developers that deploy Spring apps to PCF, you can enter Spring-specific HTTP headers.



27. If you expect requests larger than the default maximum of 16 Kbytes, enter a new value (in bytes) for **HAProxy Request Max Buffer Size**. You may need to do this, for example, to support apps that embed a large cookie or query string values in headers.

28. If your PCF deployment uses HAProxy and you want it to receive traffic only from specific sources, use the following fields:
    - **Protected Domains:** Enter a comma-separated list of domains from which PCF can receive traffic.
    - **Trusted CIDRs:** Optionally, enter a space-separated list of CIDRs to limit which IP addresses from the **Protected Domains** can send traffic to PCF.



29. The **Loggregator Port** defaults to `443` if left blank. Enter a new value to override the default.

30. For **Container Network Plugin Interface**, select one of the following:
    - **Silk**: This option is the default Container Network Plugin Interface for PAS.
    - **External**: This option exists to support NSX-T integration for vSphere deployments. Select this option only if you have deployed the NSX-T tile

for PCF.

Warning: The NSX-T integration only works for fresh installs of PCF. You cannot upgrade an existing PAS deployment to use NSX-T, and there is no upgrade path from NSX-V to NSX-T. Additionally, if your deployment is already running Silk, you cannot change it to run an additional CNI.

31. If you selected **Silk** in the previous step, review the following fields:
    a. (Optional) You can change the value in the **Applications Network Maximum Transmission Unit (MTU)** field. Pivotal recommends setting the MTU value for your application network to `1454`. Some configurations, such as networks that use GRE tunnels, may require a smaller MTU value.
    b. (Optional) Enter an IP range for the overlay network in the **Overlay Subnet** box. If you do not set a custom range, Ops Manager uses `10.255.0.0/16`.

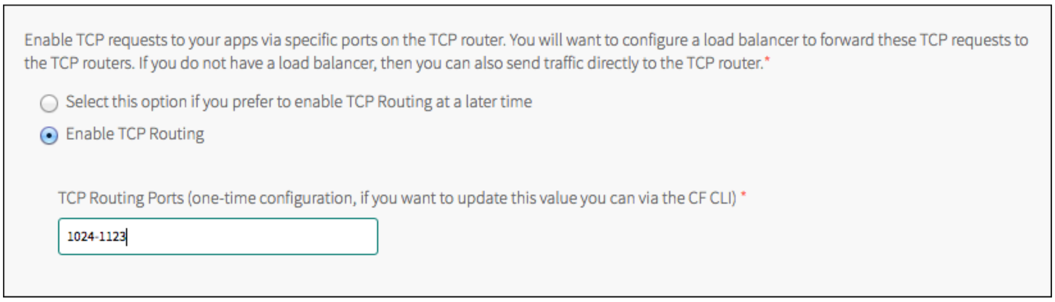
    WARNING: The overlay network IP range must not conflict with any other IP addresses in your network.

    c. Enter a UDP port number in the **VXLAN Tunnel Endpoint Port** box. If you do not set a custom port, Ops Manager uses 4789.
    d. For **Denied logging interval**, set the per-second rate limit for packets blocked by either a container-specific networking policy or by Application Security Group rules applied across the space, org, or deployment. This field defaults to `1`.
    e. For **UDP logging interval**, set the per-second rate limit for UDP packets sent and received. This field defaults to `100`.
    f. To enable logging for app traffic, select **Log traffic for all accepted/denied application packets**. See Manage Logging for Container-to-Container Networking for more information.

32. (Optional) TCP Routing is disabled by default. To enable this feature, perform the following steps:
    a. Select **Enable TCP Routing**.
    b. In **TCP Routing Ports**, enter a range of ports to be allocated for TCP Routes.

    For each TCP route you want to support, you must reserve a range of ports. This is the same range of ports you configured your load balancer with in the Pre-Deployment Steps, unless you configured DNS to resolve the TCP domain name to the TCP router directly.

    The **TCP Routing Ports** field accepts a comma-delimited list of individual ports and ranges, for example `1024-1099,30000,60000-60099`. Configuration of this field is only applied on the first deploy, and update updates to the port range are made using the cf CLI. For details about modifying the port range, see the Router Groups topic.

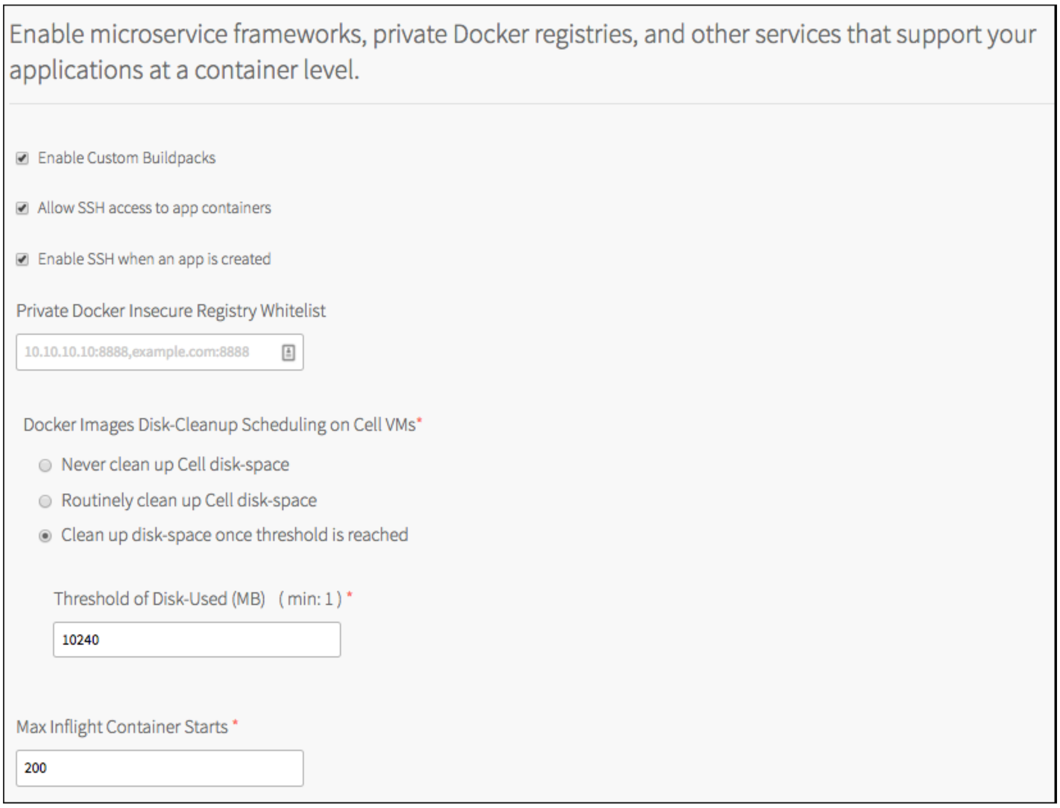


    c. Return to the top of the **Networking** screen. In **TCP Router IPs** field, make sure you have entered IP addresses within your subnet CIDR block. These will be the same IP addresses you configured your load balancer with in Pre-Deployment Steps, unless you configured DNS to resolve the TCP domain name directly to an IP you've chosen for the TCP router. You can enter multiple values as a comma-delimited list or as a range. For example, `10.254.0.1, 10.254.0.2` or `10.254.0.1-10.254.0.2`.
    d. (Optional) To disable TCP routing, click **Select this option if you prefer to enable TCP Routing at a later time** For more information, see the Configuring TCP Routing in PAS topic.

33. Click **Save**.

Step 5: Configure Application Containers

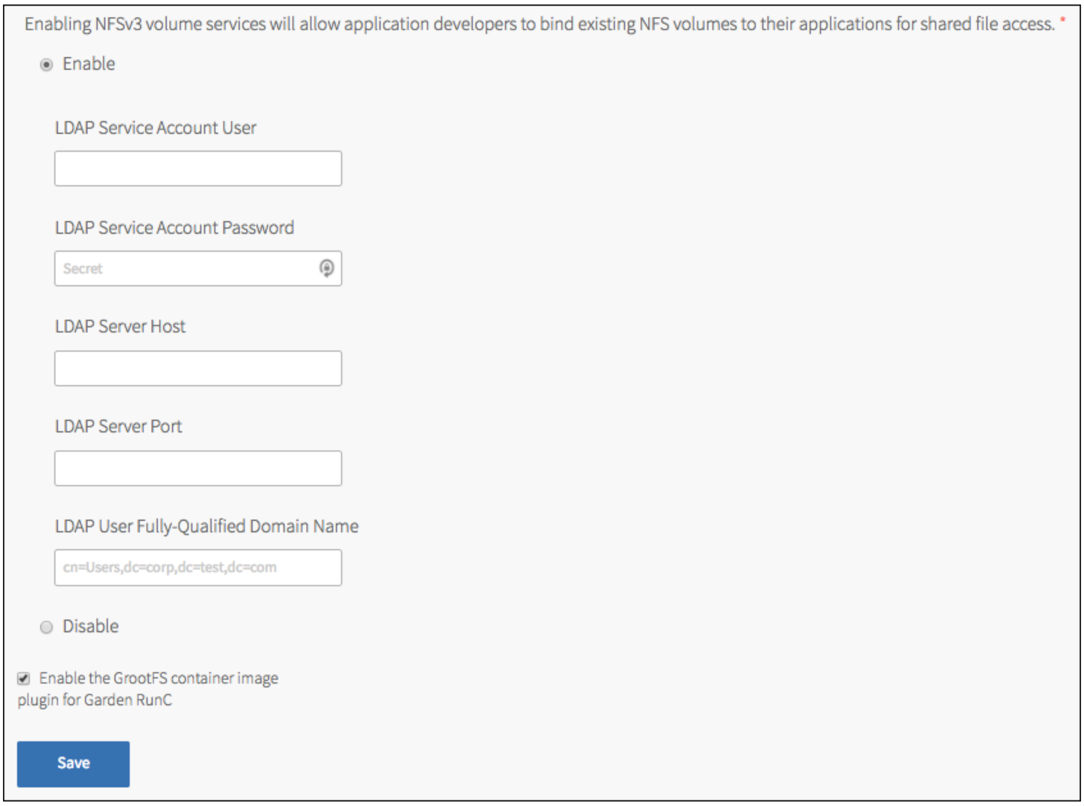
1. Select **Application Containers**.



2. The **Enable Custom Buildpacks** checkbox governs the ability to pass a custom buildpack URL to the `-b` option of the `cf push` command. By default, this ability is enabled, letting developers use custom buildpacks when deploying apps. Disable this option by disabling the checkbox. For more information about custom buildpacks, refer to the buildpacks section of the PCF documentation.
3. The **Allow SSH access to app containers** checkbox controls SSH access to application instances. Enable the checkbox to permit SSH access across your deployment, and disable it to prevent all SSH access. See the Application SSH Overview topic for information about SSH access permissions at the space and app scope.
4. If you want enable SSH access for new apps by default in spaces that allow SSH, select **Enable SSH when an app is created**. If you deselect the checkbox, developers can still enable SSH after pushing their apps by running `cf enable-ssh APP-NAME`.
5. You can configure Pivotal Application Service (PAS) to run app instances in Docker containers by supplying their IP address range(s) in the **Private Docker Insecure Registry Whitelist** textbox. See the Using Docker Registries topic for more information.
6. Select your preference for **Docker Images Disk-Cleanup Scheduling on Cell VMs**. If you choose **Clean up disk-space once threshold is reached**, enter a **Threshold of Disk-Used** in megabytes. For more information about the configuration options and how to configure a threshold, see Configuring Docker Images Disk-Cleanup Scheduling.
7. Enter a number in the **Max Inflight Container Starts** textbox. This number configures the maximum number of started instances across your deployment's Diego Cells. For more information about this feature, see Setting a Maximum Number of Started Containers.
8. Under **Enabling NFSv3 volume services**, select **Enable** or **Disable**. NFS volume services allow application developers to bind existing NFS volumes to their applications for shared file access. For more information, see the Enabling NFS Volume Services topic.

   > **Note**: In a clean install, NFSv3 volume services is enabled by default. In an upgrade, NFSv3 volume services is set to the same setting as it was in the previous deployment.

9. (Optional) To configure LDAP for NFSv3 volume services, perform the following steps:



- For **LDAP Service Account User**, enter the username of the service account in LDAP that will manage volume services.
- For **LDAP Service Account Password**, enter the password for the service account.
- For **LDAP Server Host**, enter the hostname or IP address of the LDAP server.
- For **LDAP Server Port**, enter the LDAP server port number. If you do not specify a port number, Ops Manager uses 389.
- For **LDAP Server Protocol**, enter the server protocol. If you do not specify a protocol, Ops Manager uses TCP.
- For **LDAP User Fully-Qualified Domain Name**, enter the fully qualified path to the LDAP service account. For example, if you have a service account named `volume-services` that belongs to organizational units (OU) named `service-accounts` and `my-company`, and your domain is named `domain`, the fully qualified path looks like the following:

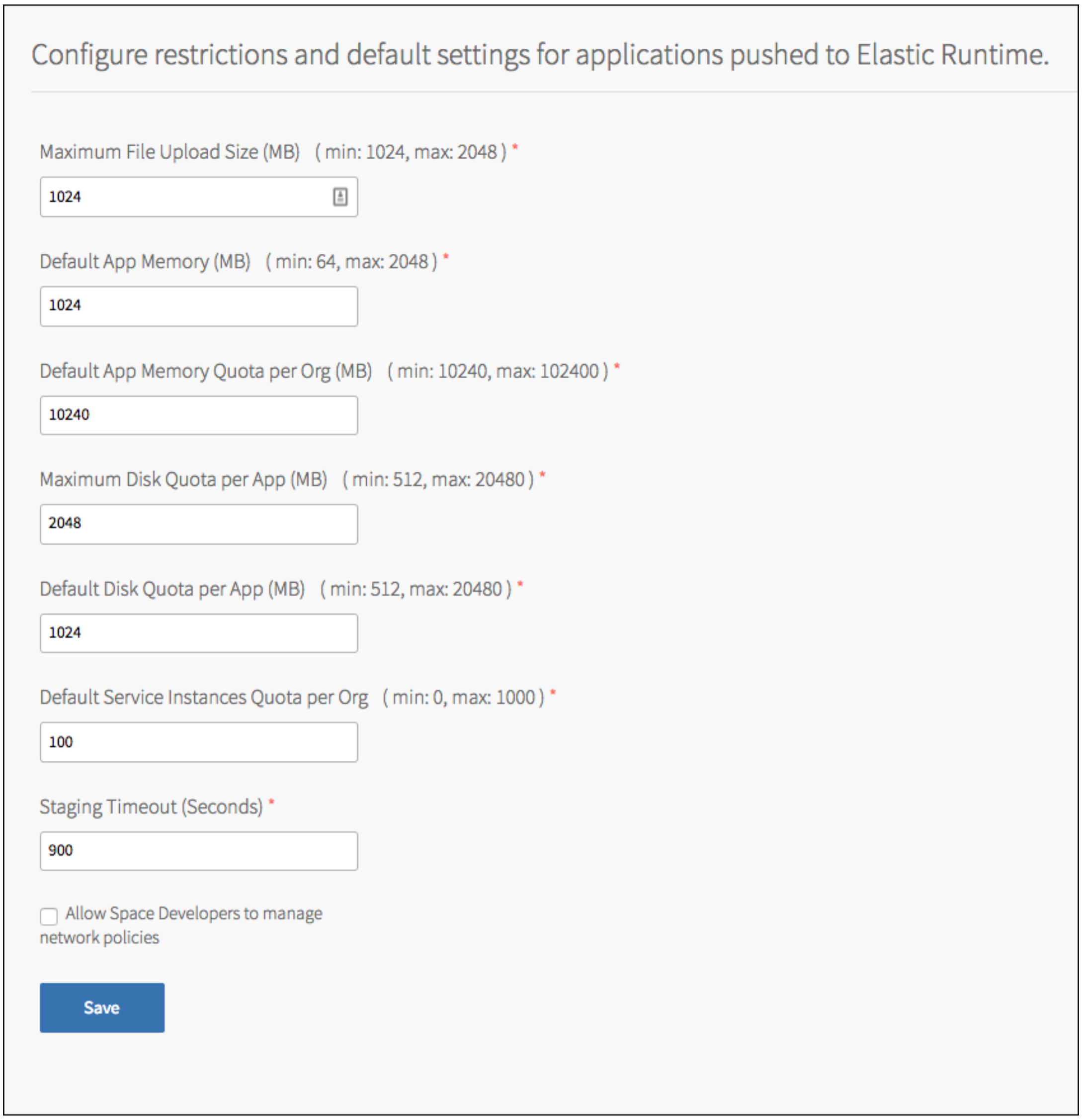
```
CN=volume-services,OU=service-accounts,OU=my-company,DC=domain,DC=com
```

10. By default, PAS manages container images using the GrootFS plugin for Garden-runC. If you experience issues with GrootFS, you can disable the plugin and use the image plugin built into Garden-runC.

11. Click **Save**.

Step 6: Configure Application Developer Controls

1. Select **Application Developer Controls**.



2. Enter the **Maximum File Upload Size (MB)**. This is the maximum size of an application upload.

3. Enter the **Default App Memory (MB)**. This is the amount of RAM allocated by default to a newly pushed application if no value is specified with the cf CLI.

4. Enter the **Default App Memory Quota per Org**. This is the default memory limit for all applications in an org. The specified limit only applies to the first installation of PAS. After the initial installation, operators can use the cf CLI to change the default value.

5. Enter the **Maximum Disk Quota per App (MB)**. This is the maximum amount of disk allowed per application.

> **Note**: If you allow developers to push large applications, PAS may have trouble placing them on Cells. Additionally, in the event of a system upgrade or an outage that causes a rolling deploy, larger applications may not successfully re-deploy if there is insufficient disk capacity. Monitor your deployment to ensure your Cells have sufficient disk to run your applications.

6. Enter the **Default Disk Quota per App (MB)**. This is the amount of disk allocated by default to a newly pushed application if no value is specified with the cf CLI.

7. Enter the **Default Service Instances Quota per Org**. The specified limit only applies to the first installation of PAS. After the initial installation, operators can use the cf CLI to change the default value .

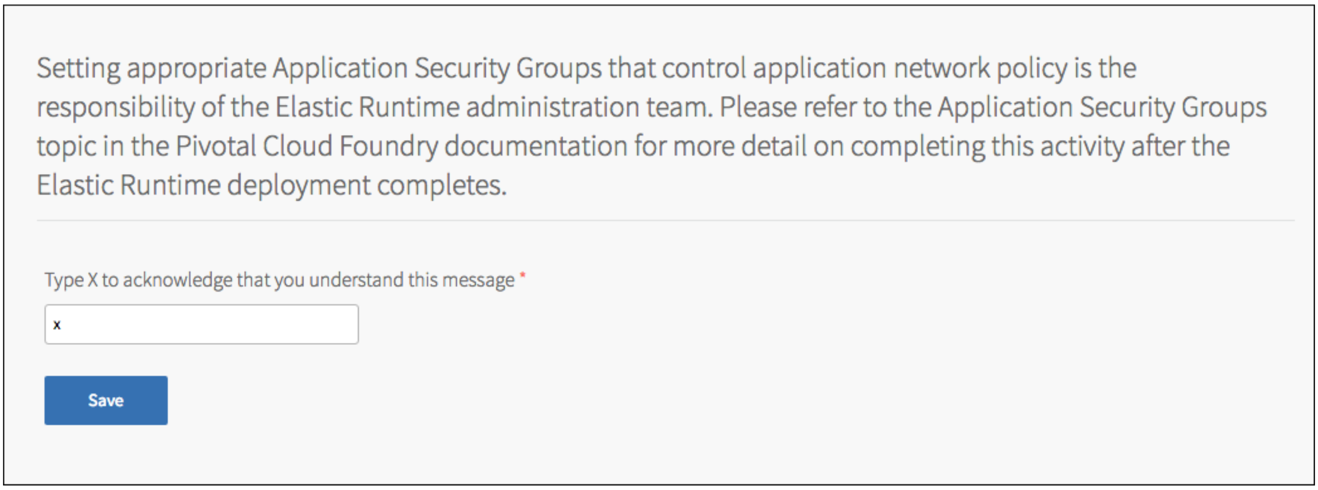
8. Enter the **Staging Timeout (Seconds)**. When you stage an application droplet with the Cloud Controller, the server times out after the number of seconds you specify in this field.

9. Select the **Allow Space Developers to manage network policies** checkbox to permit developers to manage their own network policies for their applications.

10. Click **Save**.

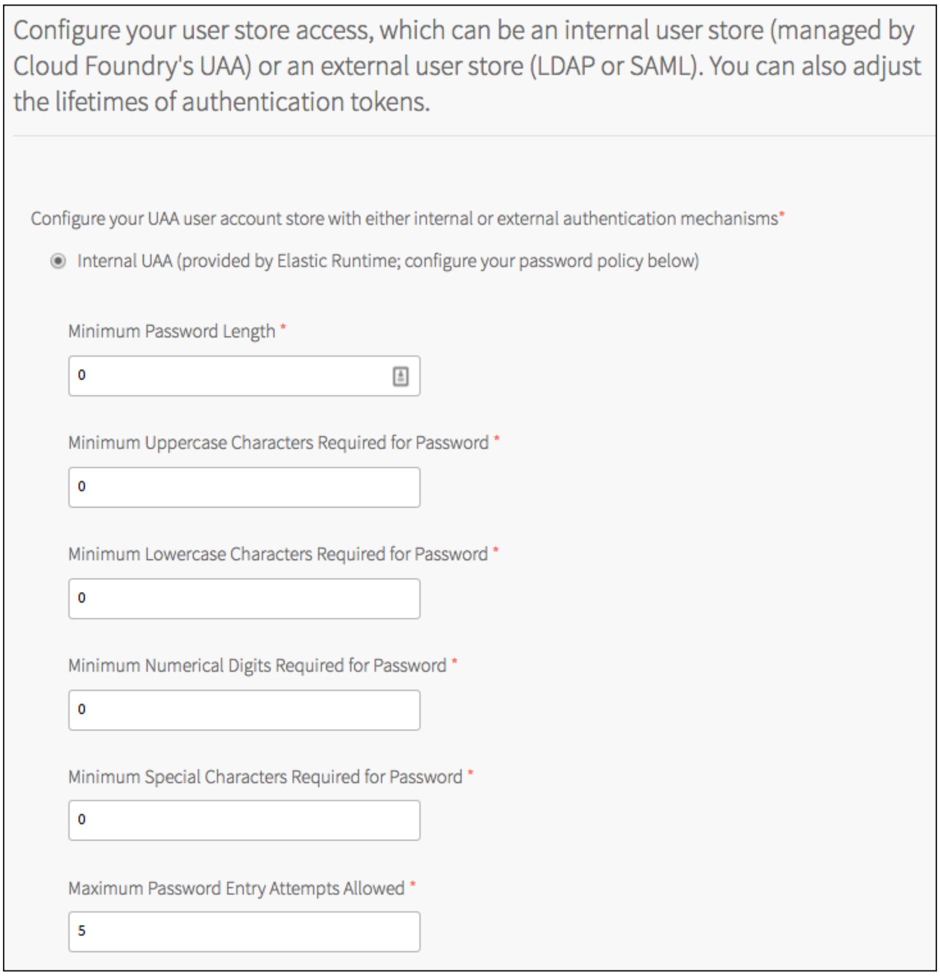
Step 7: Review Application Security Group

Setting appropriate Application Security Groups is critical for a secure deployment. Type `X` in the box to acknowledge that once the Pivotal Application Service (PAS) deployment completes, you will review and set the appropriate application security groups. See Restricting App Access to Internal PCF Components for instructions.



Step 8: Configure Authentication and Enterprise SSO

1. Select **Authentication and Enterprise SSO**.



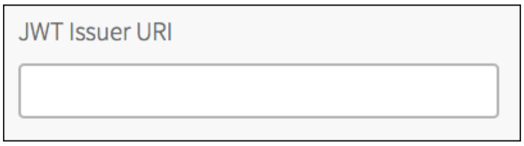
2. To authenticate user sign-ons, your deployment can use one of three types of user database: the UAA server's internal user store, an external SAML identity provider, or an external LDAP server.
   - To use the internal UAA, select the **Internal** option and follow the instructions in the Configuring UAA Password Policy topic to configure your password policy.
   - To connect to an external identity provider through SAML, scroll down to select the **SAML Identity Provider** option and follow the instructions in the Configuring PCF for SAML section of the *Configuring Authentication and Enterprise SSO for Pivotal Application Service (PAS)* topic.
   - To connect to an external LDAP server, scroll down to select the **LDAP Server** option and follow the instructions in the Configuring LDAP section of the *Configuring Authentication and Enterprise SSO for PAS* topic.

3. Click **Save**.

Step 9: Configure UAA

1. Select **UAA**.

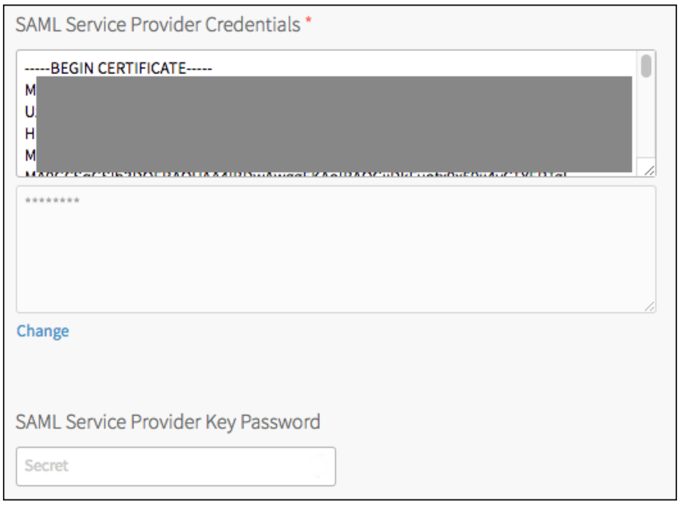
2. (Optional) Under **JWT Issuer URI**, enter the URI that UAA uses as the issuer when generating tokens.



3. Under **SAML Service Provider Credentials**, enter a certificate and private key to be used by UAA as a SAML Service Provider for signing outgoing SAML authentication requests. You can provide an existing certificate and private key from your trusted Certificate Authority or generate a self-signed certificate. The following domain must be associated with the certificate: `*.login.YOUR-SYSTEM-DOMAIN`.

   Note: The Pivotal Single Sign-On Service and Pivotal Spring Cloud Services tiles require the `*.login.YOUR-SYSTEM-DOMAIN`.

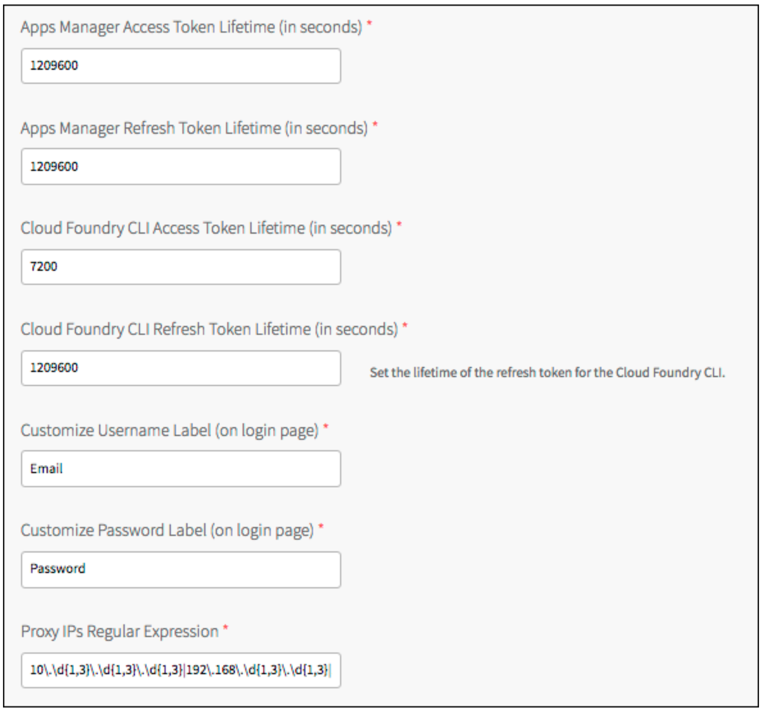
4. If the private key specified under **Service Provider Credentials** is password-protected, enter the password under **SAML Service Provider Key**



Password.

5. For **Signature Algorithm**, choose an algorithm from the dropdown menu to use for signed requests and assertions. The default value is `SHA256`.

6. (Optional) In the **Apps Manager Access Token Lifetime**, **Apps Manager Refresh Token Lifetime**, **Cloud Foundry CLI Access Token Lifetime**, and **Cloud Foundry CLI Refresh Token Lifetime** fields, change the lifetimes of tokens granted for Apps Manager and Cloud Foundry Command Line Interface (cf CLI) login access and refresh. Most deployments use the defaults.



7. (Optional) Customize the text prompts used for username and password from the cf CLI and Apps Manager login popup by entering values for **Customize Username Label (on login page)** and **Customize Password Label (on login page)**.

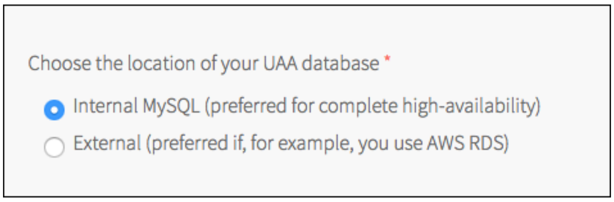
8. (Optional) The **Proxy IPs Regular Expression** field contains a pipe-delimited set of regular expressions that UAA considers to be reverse proxy IP addresses. UAA respects the `x-forwarded-for` and `x-forwarded-proto` headers coming from IP addresses that match these regular expressions. To configure UAA to respond properly to Gorouter or HAProxy requests coming from a public IP address, append a regular expression or regular expressions to match the public IP address.

9. You can configure UAA to use the internal MySQL database provided with PCF, or you can configure an external database provider. Follow the procedures in either the Internal Database Configuration or the External Database Configuration section below.

Note: If you are performing an upgrade, do not modify your existing internal database configuration or you may lose data. You must migrate your existing data before changing the configuration. See Upgrading Pivotal Cloud Foundry for additional upgrade information, and contact Pivotal Support for help.

Internal Database Configuration

1. Select **Internal MySQL**.

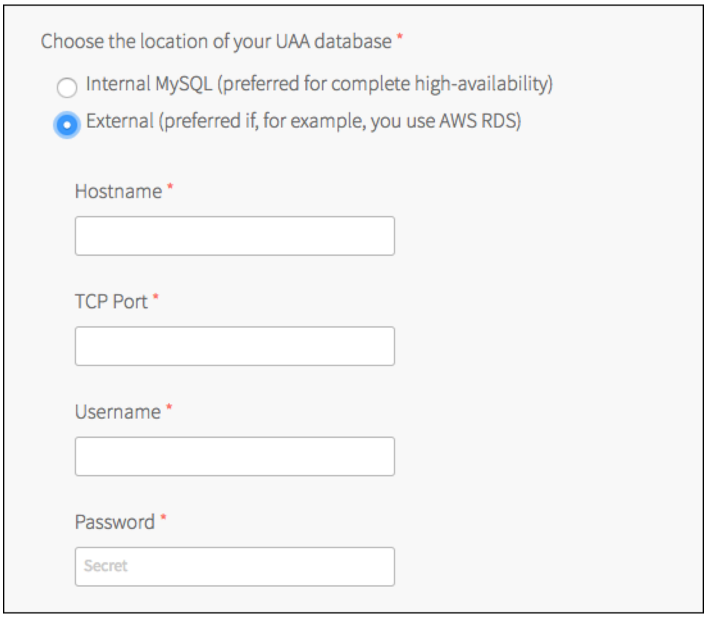


2. Click **Save**.

3. Ensure that you complete the “Configure Internal MySQL” step later in this topic to configure high availability and automatic backups for your internal MySQL databases.

External Database Configuration

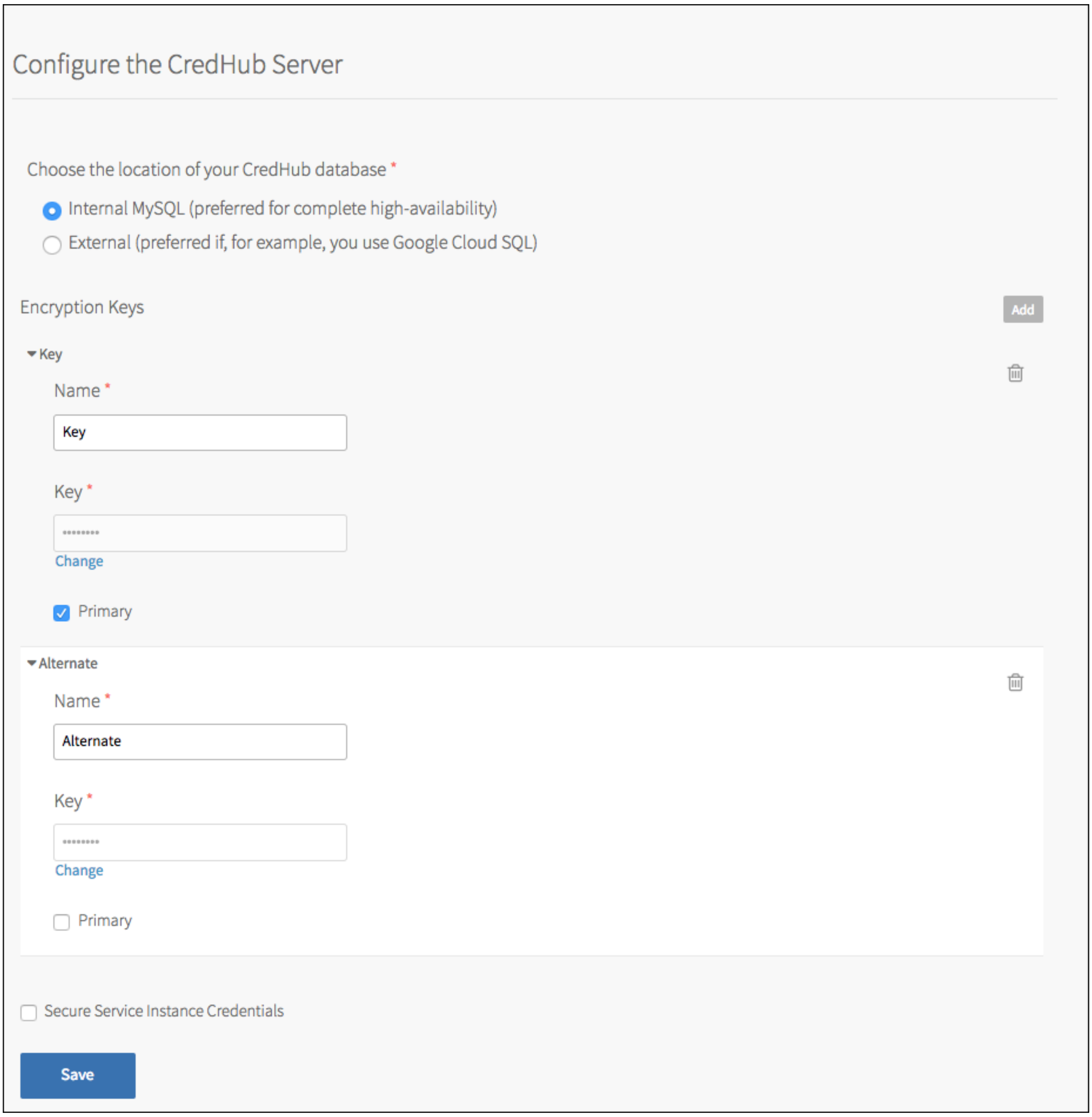
1. From the **UAA** section in Pivotal Application Service (PAS), select **External**.



2. For **Hostname**, enter the hostname of the database server.
3. For **TCP Port**, enter the port of the database server.
4. For **User Account and Authentication database username**, specify a unique username that can access this specific database on the database server.
5. For **User Account and Authentication database password**, specify a password for the provided username.
6. Click **Save**.

Step 10: (Optional) Configure CredHub

1. Select **Credhub**.



2. Choose the location of your CredHub Database. PAS includes this CredHub database for services to store their service instance credentials.

    a. If you chose **External**, enter the following:
        - **Hostname**: The IP address of your database server.
        - **TCP Port**: The port of your database server, such as `3306`.
        - **Username**: A unique username that can access this specific database on the database server.
        - **Password**: The password for the provided username.
        - **Database CA Certificate**: Enter a certificate to use for encrypting traffic to and from the database.

3. Under **Encryption Keys**, specify a key to use for encrypting and decrypting the values stored in the CredHub database.
    - **Name**: Enter the name of the key.
    - **Key**: Enter a key that is at least 20 characters in length.
    - **Primary**: Select this checkbox to use this key as your primary key.

    > **Note**: Ensure that you only mark one key as **Primary**. The UI includes an **Add** button to add more keys to support key rotation. For more information, see the Rotating PAS CredHub Encryption Keys topic.

4. If your deployment uses any PCF services that support storing service instance credentials in CredHub and you want to enable this feature, select the **Secure Service Instance Credentials** checkbox.

5. Select the **Resource Config** pane.

6. Under the **Job** column of the **CredHub** row, set the number of instances to `2`. This is the minimum instance count required for high availability.

Note: To use the Runtime CredHub feature, you must follow the additional steps in Securing Service Instance Credentials with Runtime CredHub.

Step 11: Configure System Databases

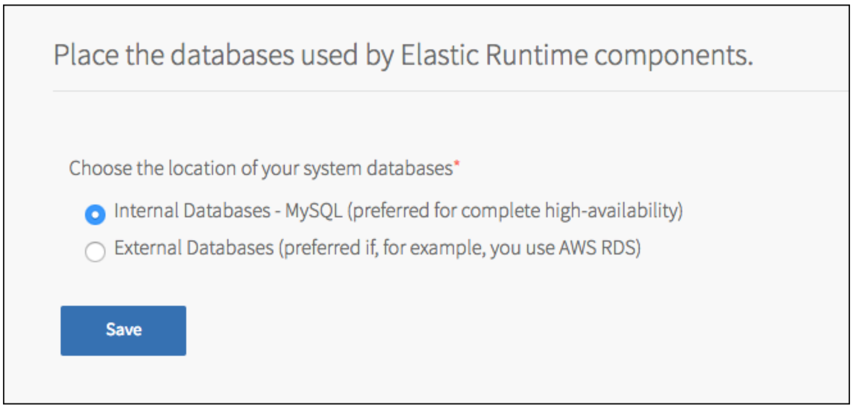
You can configure PAS to use the internal MySQL database provided with PCF, or you can configure an external database provider for the databases required by PAS.

Note: If you are performing an upgrade, do not modify your existing internal database configuration or you may lose data. You must migrate your existing data first before changing the configuration. Contact Pivotal Support for help. See Upgrading Pivotal Cloud Foundry for additional upgrade information.

Internal Database Configuration

If you want to use internal databases for your deployment, perform the following steps:

1. Select **Databases**.



2. Select **Internal Databases - MySQL**.
3. Click **Save**.

Then proceed to Step 12: (Optional) Configure Internal MySQL to configure high availability and automatic backups for your internal MySQL databases.

External Database Configuration

Note: To configure an external database for UAA, see the *External Database Configuration* section of Configure UAA.

Note: The exact procedure to create databases depends upon the database provider you select for your deployment. The following procedure uses AWS RDS as an example. You can configure a different database provider that provides MySQL support, such as Google Cloud SQL.

Warning: Protect whichever database you use in your deployment with a password.

To create your Pivotal Application Service (PAS) databases, perform the following steps:

1. Add the `ubuntu` account key pair from your IaaS deployment to your local SSH profile so you can access the Ops Manager VM. For example, in AWS, you add a key pair created in AWS:

```
$ ssh-add aws-keypair.pem
```

2. SSH in to your Ops Manager using the Ops Manager FQDN and the username `ubuntu`:

```
$ ssh ubuntu@OPS-MANAGER-FQDN
```

3. Log in to your MySQL database instance using the appropriate hostname and user login values configured in your IaaS account. For example, to log in to your AWS RDS instance, run the following MySQL command:

```
$ mysql --host=RDSHOSTNAME --user=RDSUSERNAME --password=RDSPASSWORD
```

4. Run the following MySQL commands to create databases for the eleven PAS components that require a relational database:

```
CREATE database ccdb;
CREATE database notifications;
CREATE database autoscale;
CREATE database app_usage_service;
CREATE database routing;
CREATE database diego;
CREATE database account;
CREATE database nfsvolume;
CREATE database networkpolicyserver;
CREATE database silk;
CREATE database locket;
```

5. Type `exit` to quit the MySQL client, and `exit` again to close your connection to the Ops Manager VM.

6. In PAS, select **Databases**.

7. Select the **External Databases** option.

8. For **Hostname**, enter the hostname of the database server.

9. For **TCP Port**, enter the port of the database server.

10. Each component that requires a relational database has two corresponding fields: one for the database username and one for the database password. For each set of fields, specify a unique username that can access this specific database on the database server and a password for the provided username.

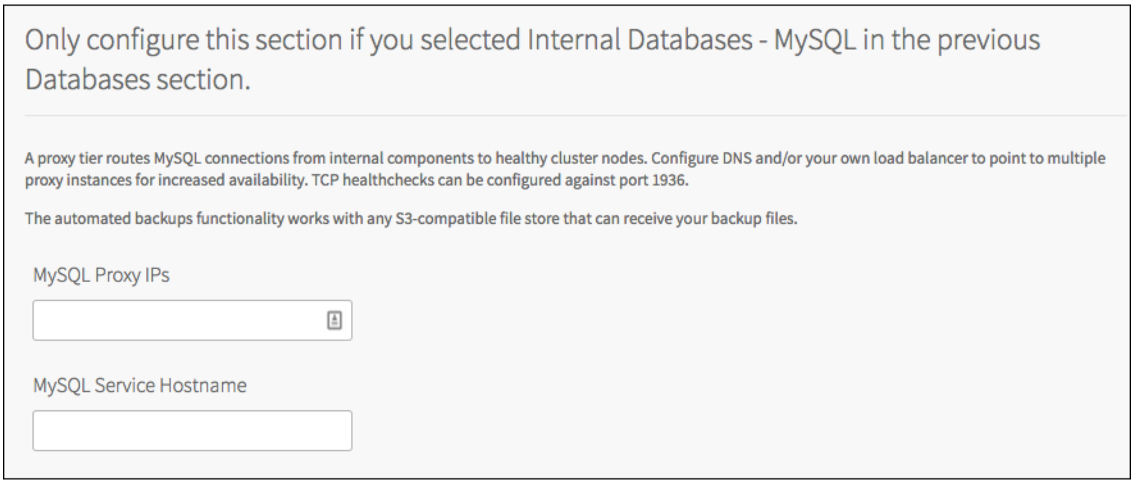
11. Click **Save**.

Step 12: (Optional) Configure Internal MySQL

Note: You only need to configure this section if you have selected **Internal Databases - MySQL** in the **Databases** section.

1. Select **Internal MySQL**.

2. In the **MySQL Proxy IPs** field, enter one or more comma-delimited IP addresses that are not in the reserved CIDR range of your network. If a MySQL node fails, these proxies re-route connections to a healthy node. See the [MySQL Proxy](#) topic for more information.



3. For **MySQL Service Hostname**, enter an IP address or hostname for your load balancer. If a MySQL proxy fails, the load balancer re-routes connections to a healthy proxy. If you leave this field blank, components are configured with the IP address of the first proxy instance entered above.

> **Warning:** You must configure a load balancer to achieve complete high-availability.

4. In the **Replication canary time period** field, leave the default of 30 seconds or modify the value based on the needs of your deployment. Lower numbers cause the canary to run more frequently, which means that the canary reacts more quickly to replication failure but adds load to the database.

5. In the **Replication canary read delay** field, leave the default of 20 seconds or modify the value based on the needs of your deployment. This field configures how long the canary waits, in seconds, before verifying that data is replicating across each MySQL node. Clusters under heavy load can experience a small replication lag as write-sets are committed across the nodes.

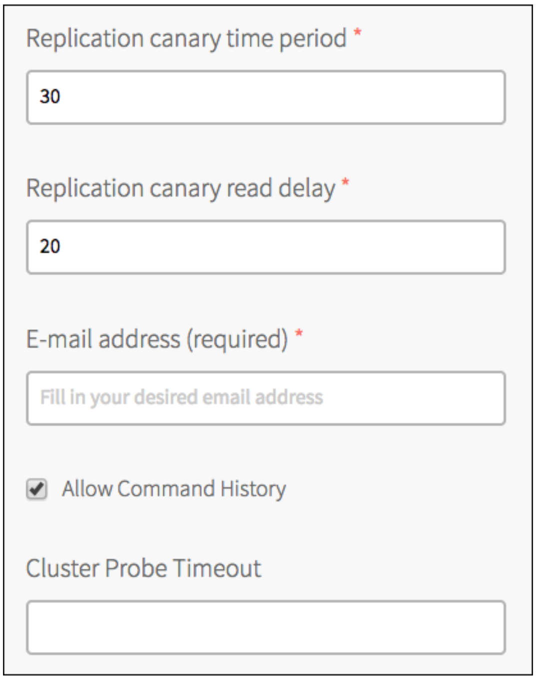
6. (**Required**): In the **E-mail address** field, enter the email address where the MySQL service sends alerts when the cluster experiences a replication issue or when a node is not allowed to auto-rejoin the cluster.

7. To prohibit the creation of command line history files on the MySQL nodes, disable the **Allow Command History** checkbox.

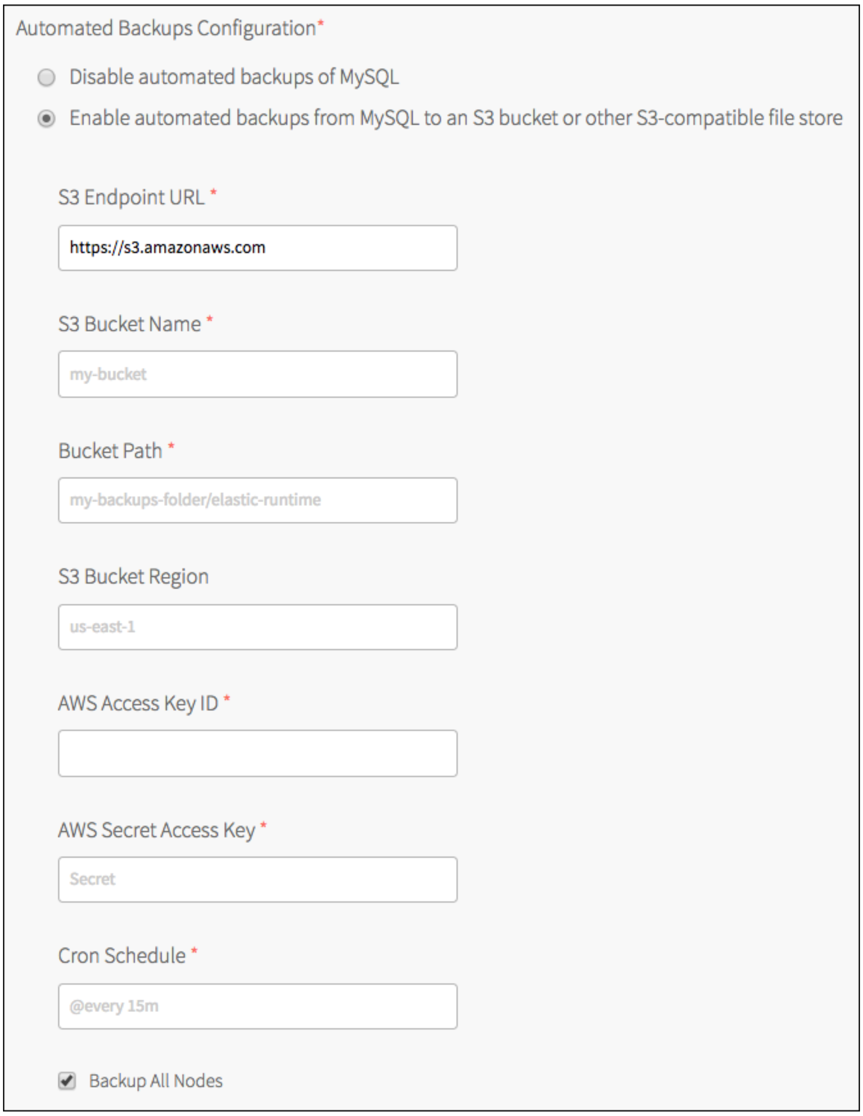
8. To allow the admin and roadmin to connect from any remote host, enable the **Allow Remote Admin Access** checkbox. When the checkbox is disabled, admins must `bosh ssh` into each MySQL VM to connect as the MySQL super user.

> **Note:** Network configuration and Application Security Groups restrictions may still limit a client's ability to establish a connection with the databases.

9. For **Cluster Probe Timeout**, enter the maximum amount of time, in seconds, that a new node will search for existing cluster nodes. If left blank, the default value is 10 seconds.



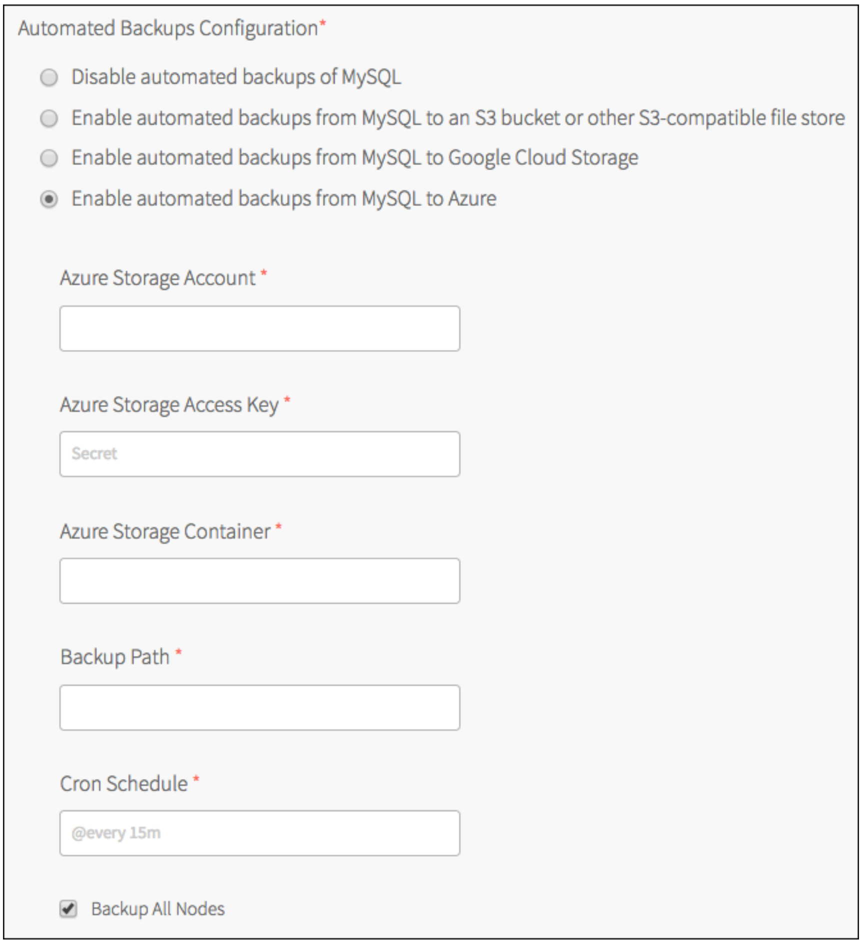
10. Under **Automated Backups Configuration**, choose one of three options for MySQL backups:
    - **Disable automatic backups of MySQL** disables automatic backups, but you can still deploy the Backup Prepare Node if you use BOSH Backup and Restore to back up your MySQL database. For more information, see the Backing Up Pivotal Cloud Foundry with BBR topic.
    - **Enable automated backups from MySQL to an S3 bucket or other S3-compatible file store** saves your backups to an existing Amazon Web Services (AWS) or Ceph S3-compatible blobstore.



This option requires the following fields:

- For **S3 Bucket Name**, enter the name of your S3 bucket. Do not include an `s3://` prefix, a trailing `/`, or underscores. If the bucket does not already exist, it will be created automatically.
- For **Bucket Path**, specify a folder within the bucket to hold your MySQL backups. Do not include a trailing `/`.
- For **S3 Bucket Region**, enter the AWS region where the bucket is located, such as `us-east-1`.
- For **AWS Access Key ID** and **AWS Secret Access Key**, enter your AWS or Ceph credentials.
- For **Cron Schedule**, enter a valid cron expression to schedule your automated backups. Cron uses your computer's local time zone.
- Enable **Backup All Nodes** to make unique backups from each instance of the MySQL server rather than just the first MySQL server instance.

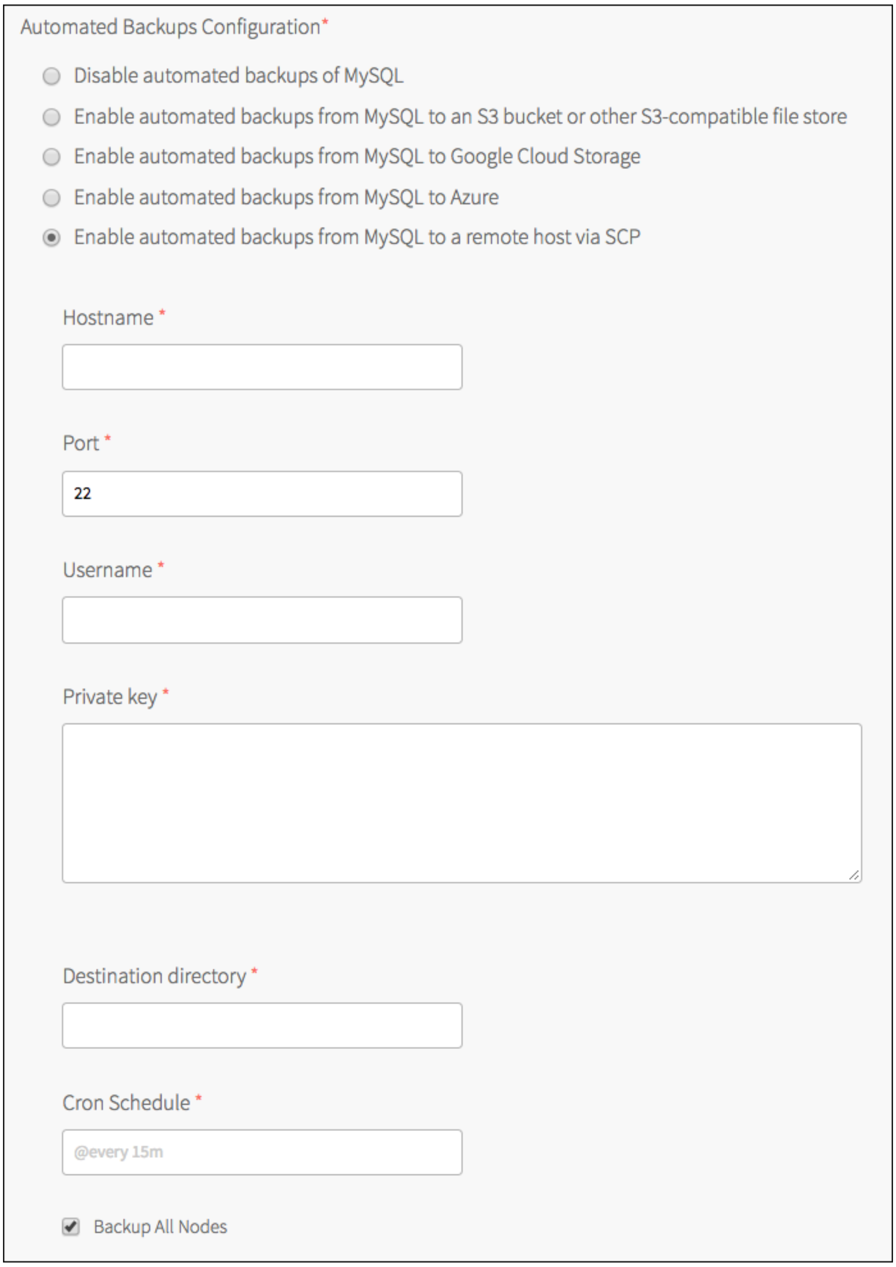
- **Enable automated backups from MySQL to Azure** saves your backups to Azure.



This option requires the following fields:

- For **Azure Storage Account**, enter the name of an existing Azure storage account where backups will be uploaded. For more information about creating and managing an Azure storage account, see the Azure documentation.
- For **Azure Storage Access Key**, enter an Azure storage access key for the storage account.
- For **Azure Storage Container**, enter the name of an existing Azure storage container that will store the backups.
- For **Backup Path**, enter the path within the Azure storage container where backups will be uploaded.
- For **Cron Schedule**, enter a valid cron expression to schedule your automated backups. Cron uses your computer's local time zone.
- Enable **Backup All Nodes** to make unique backups from each instance of the MySQL server rather than just the first MySQL server instance.

- **Enable automated backups from MySQL to a remote host via SCP**saves your backups to a remote host using secure copy protocol (SCP).



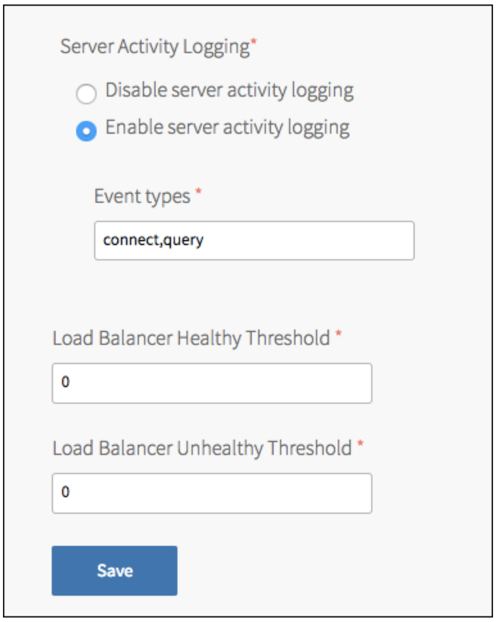
This option requires the following fields:

- For **Hostname**, enter the name of your SCP host.
- For **Port**, enter your SCP port. This should be the TCP port that your SCP host uses for SSH. The default port is `22`.
- For **Username**, enter your SSH username for the SCP host.
- For **Private key**, paste in your SSH private key.
- For **Destination directory**, enter the directory on the SCP host where you want to save backup files.
- For **Cron Schedule**, enter a valid cron expression to schedule your automated backups. Cron uses your computer's local time zone.
- Enable **Backup All Nodes** to make unique backups from each instance of the MySQL server rather than just the first MySQL server instance.

> **Note**: If you choose to enable automated MySQL backups, set the number of instances for the **Backup Prepare Node** under the **Resource Config** section of the Pivotal Application Service (PAS) tile to `1`.

11. If you want to log audit events for internal MySQL, select **Enable server activity logging** under **Server Activity Logging**.

    a. For the **Event types** field, you can enter the events you want the MySQL service to log. By default, this field includes `connect` and `query`, which tracks who connects to the system and what queries are processed. For more information, see the Logging Events section of the MariaDB documentation.



12. Enter values for the following fields:
    - **Load Balancer Healthy Threshold**: Specifies the amount of time, in seconds, to wait until declaring the MySQL proxy instance started. This allows an external load balancer time to register the instance as healthy.
    - **Load Balancer Unhealthy Threshold**: Specifies the amount of time, in seconds, that the MySQL proxy continues to accept connections before shutting down. During this period, the healthcheck reports as unhealthy to cause load balancers to fail over to other proxies. You must enter a value greater than or equal to the maximum time it takes your load balancer to consider a proxy instance unhealthy, given repeated failed healthchecks.

13. If you want to enable the MySQL interruptor feature, select the checkbox to **Prevent node auto re-join**. This feature stops all writes to the MySQL database if it notices an inconsistency in the dataset between the nodes. For more information, see the Interruptor section in the MySQL for PCF documentation.

14. Click **Save**.

Step 13: Configure File Storage

For production-level PCF deployments on vSphere, the recommended selection is **External S3-Compatible** and the use of an external filestore. For more information about production-level PCF deployments on vSphere, see the Reference Architecture for Pivotal Cloud Foundry on vSphere.

For more factors to consider when selecting file storage, see Considerations for Selecting File Storage in Pivotal Cloud Foundry.

Internal Filestore

Internal file storage is only appropriate for small, non-production deployments.

To use the PCF internal filestore, perform the following steps:

1. In the Pivotal Application Service (PAS) tile, select **File Storage**.

2. Select **Internal WebDAV**, and click **Save**.

External S3 or Ceph Filestore

To use an external S3-compatible filestore for your Pivotal Application Service (PAS) file storage, perform the following steps:

1. In the PAS tile, select **File Storage**.

2. Select the **External S3-Compatible Filestore** option and complete the following fields:
    - Prepend `https://` to the endpoint for your region and copy it into the Ops Manager **URL Endpoint** field.

For example, in the **us-west-2** region, use `https://s3-us-west-2.amazonaws.com/`.
- Enter the **Access Key** and **Secret Key** of the `pcf-user` you created when configuring AWS for PCF.
- For **S3 Signature Version** and **Region**, use the **V4 Signature** values. AWS recommends using Signature Version 4.
- For **Region**, enter the region in which your S3 buckets are located, such as `us-west-2`.
- Select **Server-side Encryption (available for AWS S3 only)** to encrypt the contents of your S3 filestore.
- Enter values for the remaining fields as follows:

Ops Manager Field	Value	Description
Buildpacks Bucket Name	`pcf-buildpacks-bucket`	This S3 bucket stores app buildpacks.
Droplets Bucket Name	`pcf-droplets-bucket`	This S3 bucket stores app droplets. Pivotal recommends that you use a unique bucket name for droplets, but you can also use the same name as above.
Packages Bucket Name	`pcf-packages-bucket`	This S3 bucket stores app packages. Pivotal recommends that you use a unique bucket name for packages, but you can also use the same name as above.
Resources Bucket Name	`pcf-resources-bucket`	This S3 bucket stores app resources. Pivotal recommends that you use a unique bucket name for app resources, but you can also use the same name as above.

3. Click **Save**.

Note: For more information regarding AWS S3 Signatures, see the Authenticating Requests topic in the AWS documentation.

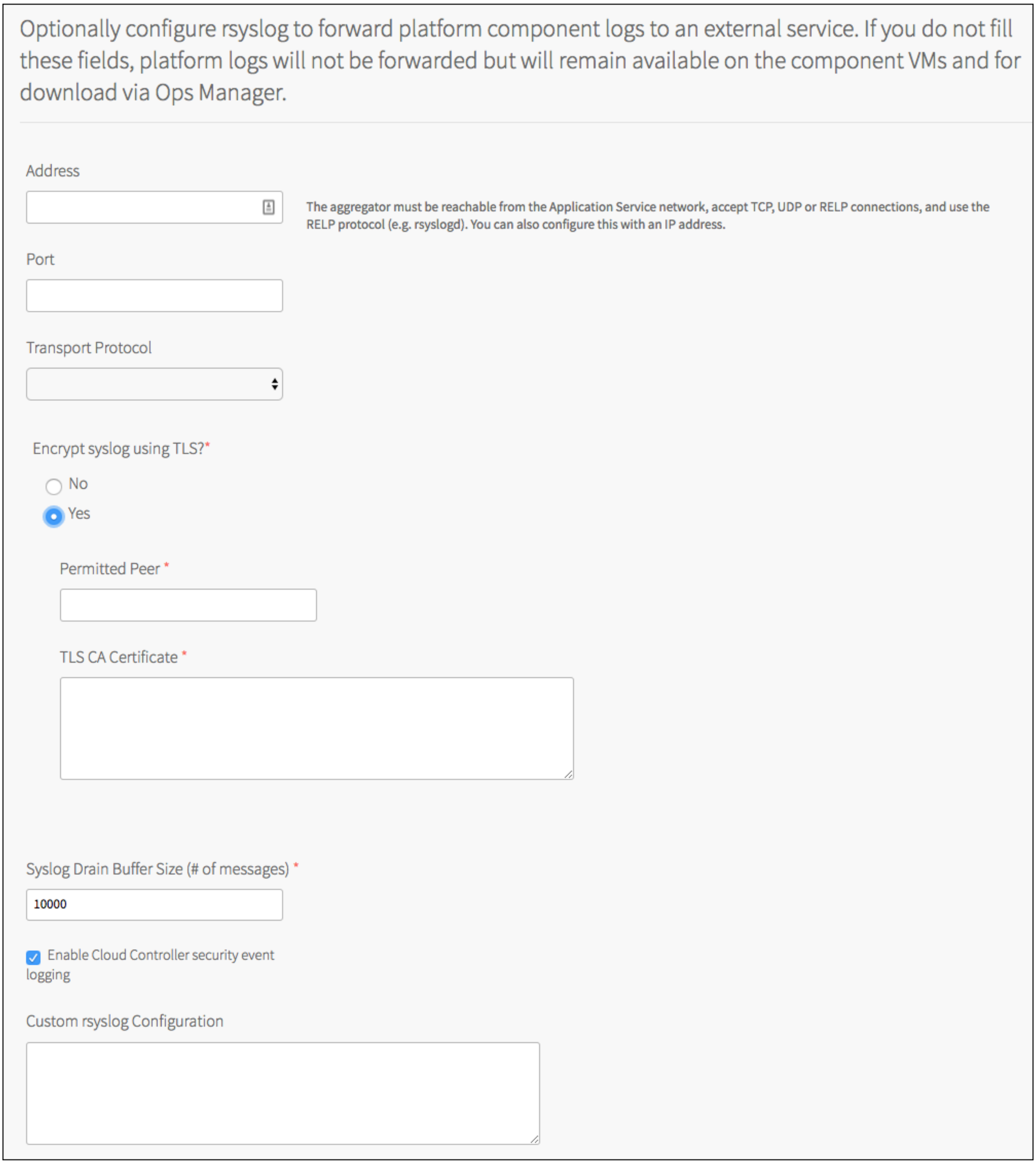
Other IaaS Storage Options

Google Cloud Storage and Azure Storage are also available as file storage options but have not been evaluated for typical PCF on vSphere installations.

Step 14: (Optional) Configure System Logging

If you forward logging messages to an external Reliable Event Logging Protocol (RELP) server, complete the following steps:

1. Select the **System Logging** section that is located within your PAS **Settings** tab.



2. Enter the IP address of your syslog server in **Address**.

3. Enter the port of your syslog server in **Port**. The default port for a syslog server is `514`.

> **Note**: The host must be reachable from the PAS network, accept TCP connections, and use the RELP protocol. Ensure your syslog server listens on external interfaces.

4. Select a **Transport Protocol** to use when forwarding logs.

5. If you plan to use TLS encryption when sending logs to the remote server, select **Yes** when answering the **Encrypt syslog using TLS?** question.

    a. In the **Permitted Peer** field, enter either the name or SHA1 fingerprint of the remote peer.
    b. In the **TLS CA Certificate** field, enter the TLS CA Certificate for the remote server.

6. For the **Syslog Drain Buffer Size**, enter the number of messages the Doppler server can hold from Metron agents before the server starts to drop them. See the Loggregator Guide for Cloud Foundry Operators topic for more details.

7. If you want to include security events in your log stream, select the **Enable Cloud Controller security event logging** checkbox. This logs all API requests, including the endpoint, user, source IP address, and request result, in the Common Event Format (CEF).

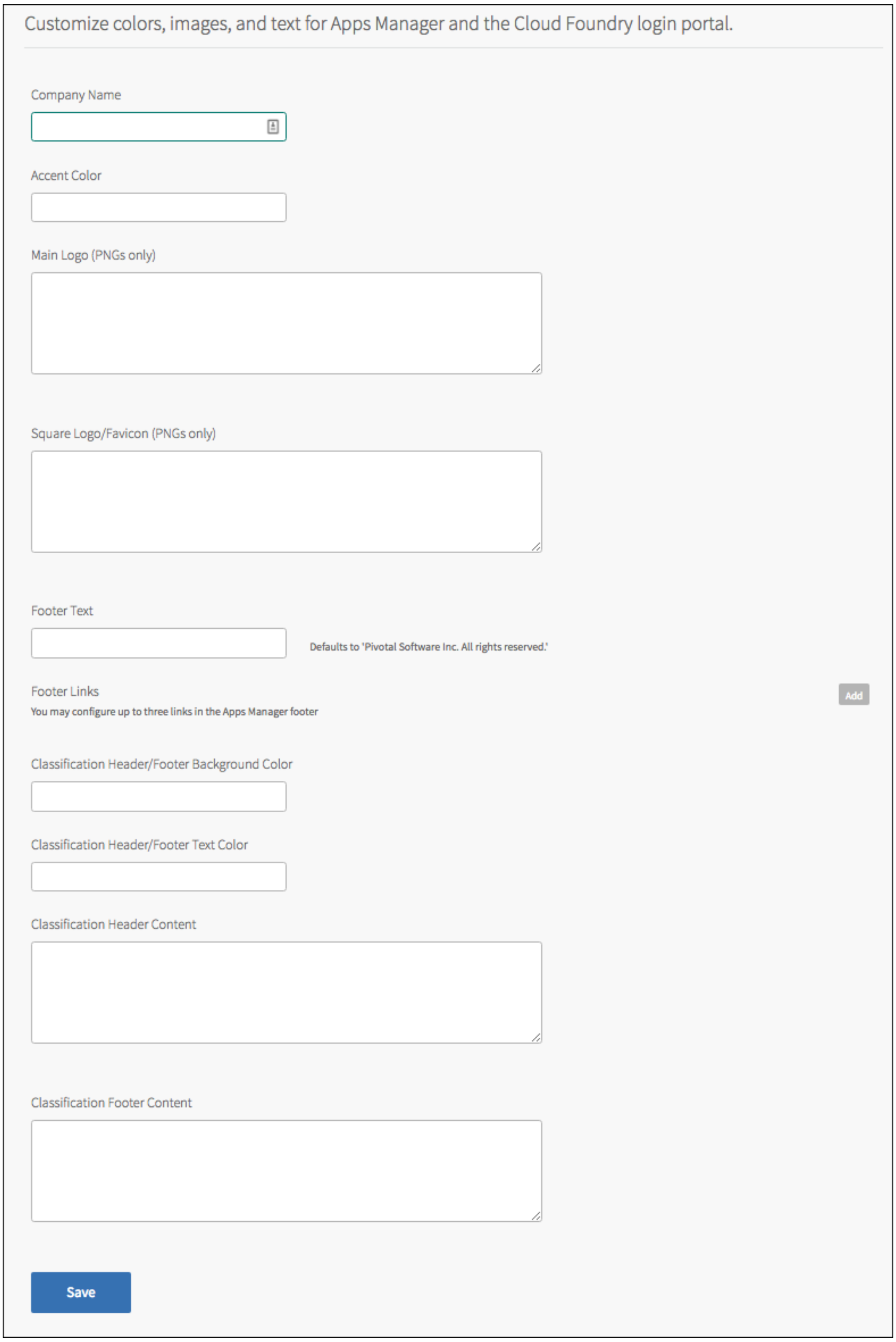
8. If you want to specify a custom syslog formatting rule, enter it in the **Custom rsyslog Configuration** field in RainerScript syntax.

9. Click **Save**.

Step 15: (Optional) Customize Apps Manager

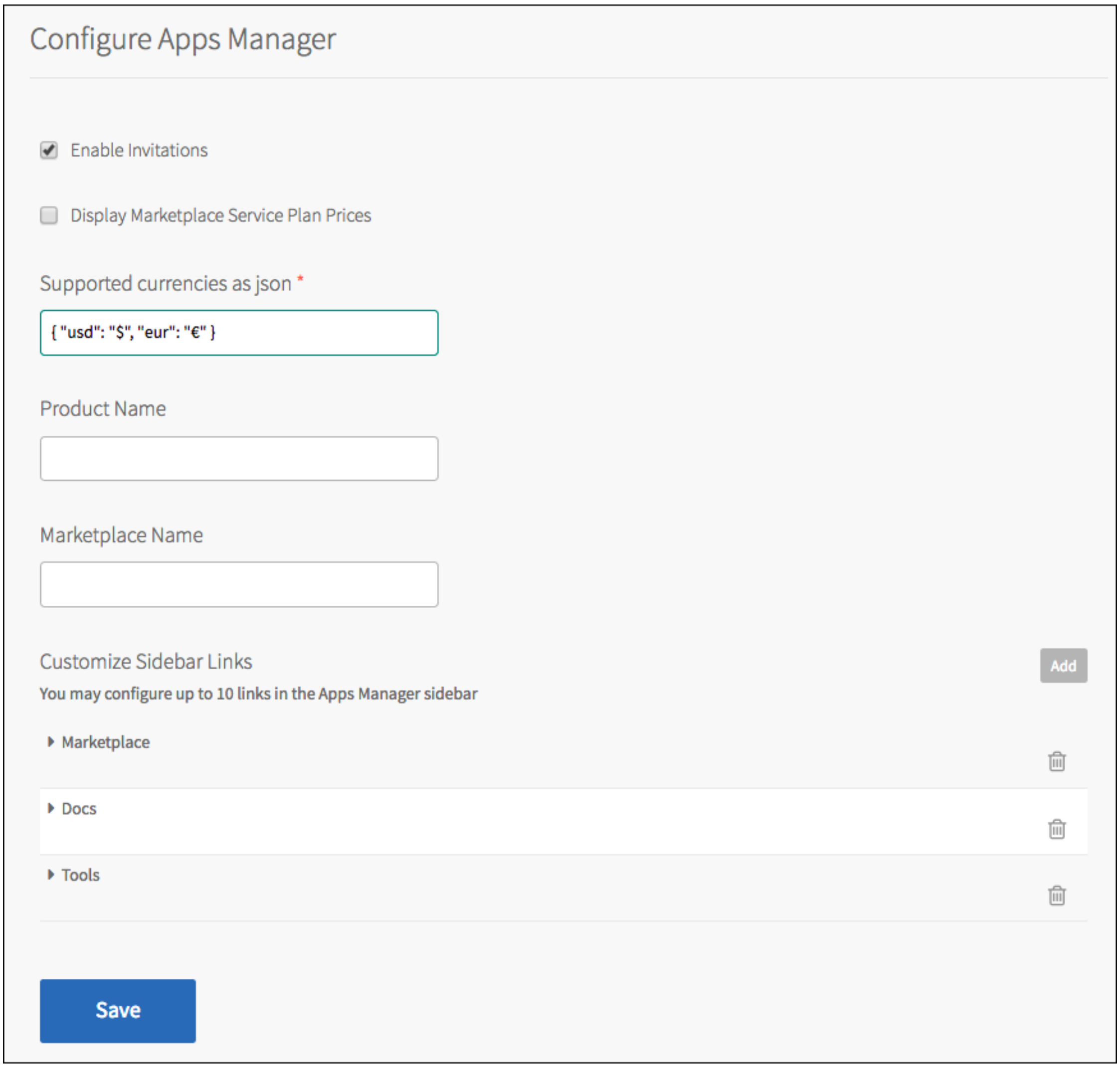
The **Custom Branding** and **Apps Manager** sections customize the appearance and functionality of Apps Manager. Refer to Custom Branding Apps Manager for descriptions of the fields on these pages and for more information about customizing Apps Manager.

1. Select **Custom Branding**. Use this section to configure the text, colors, and images of the interface that developers see when they log in, create an account, reset their password, or use Apps Manager.



2. Click **Save** to save your settings in this section.

3. Select **Apps Manager**.



4. Select **Enable Invitations** to enable invitations in Apps Manager. Space Managers can invite new users for a given space, Org Managers can invite new users for a given org, and Admins can invite new users across all orgs and spaces. See the Inviting New Users section of the *Managing User Roles with Apps Manager* topic for more information.

5. Select **Display Marketplace Service Plan Prices** to display the prices for your services plans in the Marketplace.

6. Enter the **Supported currencies as json** to appear in the Marketplace. Use the format `{"CURRENCY-CODE":"SYMBOL"}`. This defaults to `{"usd": "$", "eur": "€"}`.

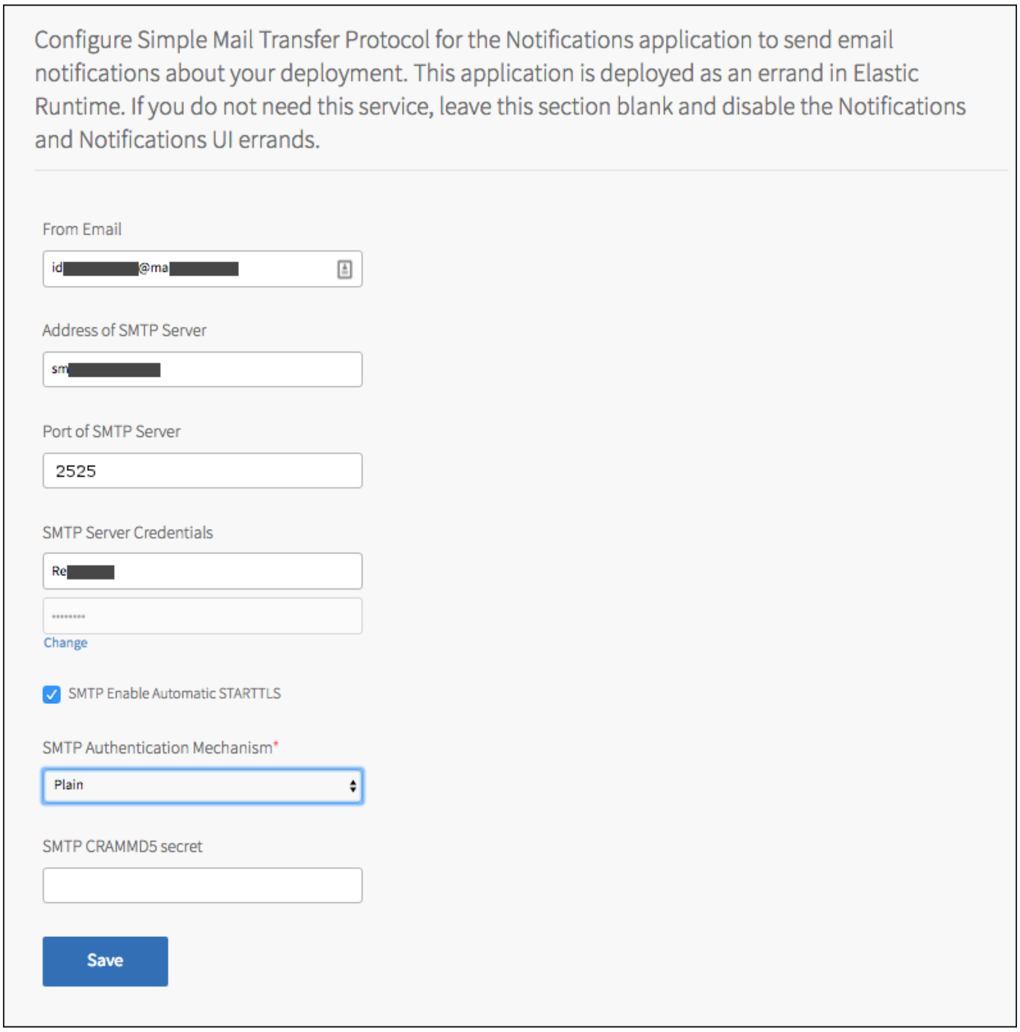
7. Use **Product Name**, **Marketplace Name**, and **Customize Sidebar Links** to configure page names and sidebar links in the **Apps Manager** and **Marketplace** pages.

8. Click **Save** to save your settings in this section.

Step 16: (Optional) Configure Email Notifications

PAS uses SMTP to send invitations and confirmations to Apps Manager users. You must complete the **Email Notifications** page if you want to enable end-user self-registration.

1. Select **Email Notifications**.



2. Enter your reply-to and SMTP email information

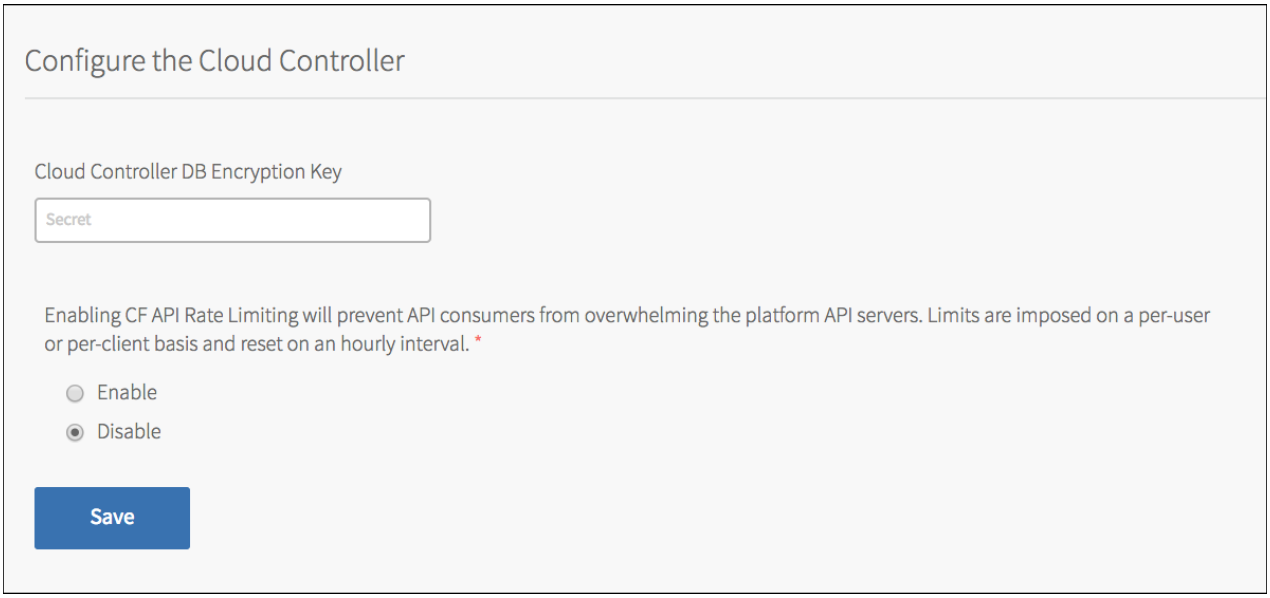
3. Verify your authentication requirements with your email administrator and use the **SMTP Authentication Mechanism** drop-down menu to select `None`, `Plain`, or `CRAMMD5`. If you have no SMTP authentication requirements, select `None`.

4. Click **Save**.

> **Note:** If you do not configure the SMTP settings using this form, the administrator must create orgs and users using the cf CLI tool. See Creating and Managing Users with the cf CLI for more information.

Step 17: Configure Cloud Controller

1. Click **Cloud Controller**.



2. Enter your **Cloud Controller DB Encryption Key** if all of the following are true:
    - You deployed Pivotal Application Service (PAS) previously.
    - You then stopped PAS or it crashed.
    - You are re-deploying PAS with a backup of your Cloud Controller database.

    See Backing Up Pivotal Cloud Foundry for more information.

3. CF API Rate Limiting prevents API consumers from overwhelming the platform API servers. Limits are imposed on a per-user or per-client basis and reset on an hourly interval.

    To disable CF API Rate Limiting, select **Disable** under **Enable CF API Rate Limiting**. To enable CF API Rate Limiting, perform the following steps:

    a. Under **Enable CF API Rate Limiting**, select **Enable**.
    b. For **General Limit**, enter the number of requests a user or client is allowed to make over an hour interval for all endpoints that do not have a custom limit. The default value is `2000`.
    c. For **Unauthenticated Limit**, enter the number of requests an unauthenticated client is allowed to make over an hour interval. The default value is `100`.

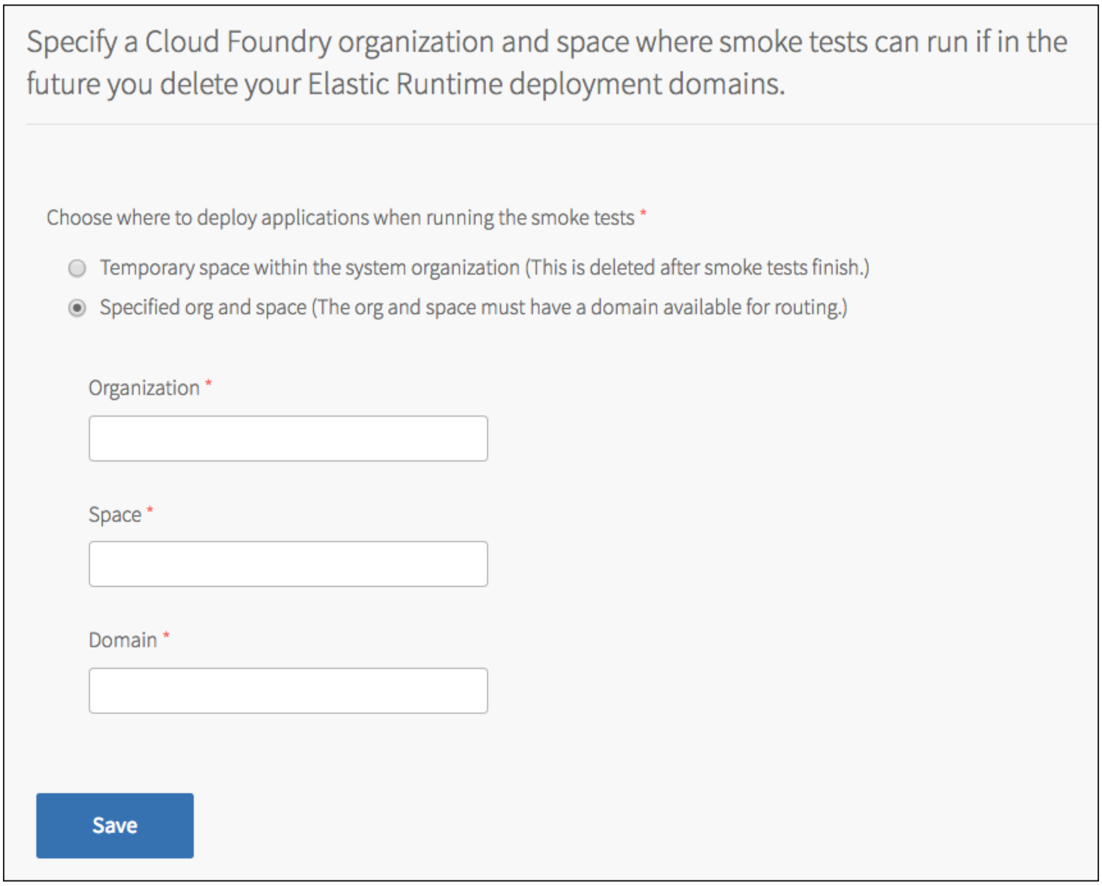
4. Click **Save**.

Step 18: Configure Smoke Tests

The Smoke Tests errand runs basic functionality tests against your Pivotal Application Service (PAS) deployment after an installation or update. In this section, choose where to run smoke tests. In the **Errands** section, you can choose whether or not to run the Smoke Tests errand.

1. Select **Smoke Tests**.

2. If you have a shared apps domain, select **Temporary space within the system organization**, which creates a temporary space within the `system` organization for running smoke tests and deletes the space afterwards. Otherwise, select **Specified org and space** and complete the fields to specify where you want to run smoke tests.



3. Click **Save**.

Step 19: (Optional) Enable Advanced Features

The **Advanced Features** section of Pivotal Application Service (PAS) includes new functionality that may have certain constraints. Although these features are fully supported, Pivotal recommends caution when using them in production environments.

Diego Cell Memory and Disk Overcommit

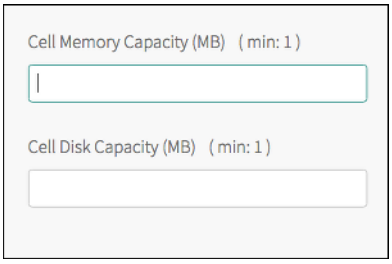
If your apps do not use the full allocation of disk space and memory set in the **Resource Config** tab, you might want use this feature. These fields control the amount to overcommit disk and memory resources to each Diego Cell VM.

For example, you might want to use the overcommit if your apps use a small amount of disk and memory capacity compared to the amounts set in the **Resource Config** settings for **Diego Cell**.

> **Note**: Due to the risk of app failure and the deployment-specific nature of disk and memory use, Pivotal has no recommendation about how much, if any, memory or disk space to overcommit.

To enable overcommit, follow these steps:

1. Select **Advanced Features**.



2. Enter the total desired amount of Diego cell memory value in the **Cell Memory Capacity (MB)** field. Refer to the **Diego Cell** row in the **Resource Config** tab for the current Cell memory capacity settings that this field overrides.

3. Enter the total desired amount of Diego cell disk capacity value in the **Cell Disk Capacity (MB)** field. Refer to the **Diego Cell** row in the **Resource Config** tab for the current Cell disk capacity settings that this field overrides.

4. Click **Save**.

> **Note**: Entries made to each of these two fields set the total amount of resources allocated, not the overage.

Whitelist for Non-RFC-1918 Private Networks

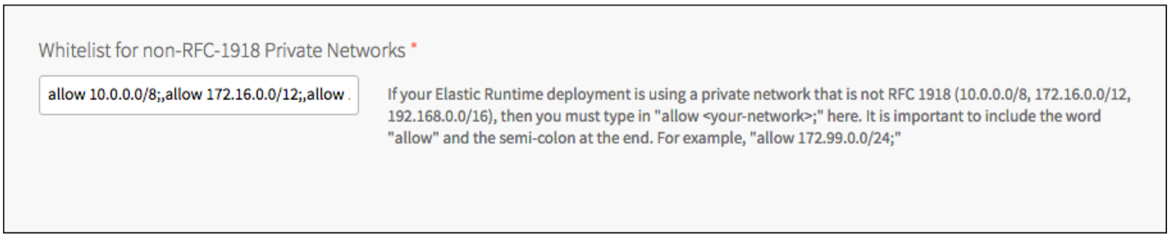
Some private networks require extra configuration so that internal file storage (WebDAV) can communicate with other PCF processes.

The **Whitelist for non-RFC-1918 Private Networks** field is provided for deployments that use a non-RFC 1918 private network. This is typically a private network other than `10.0.0.0/8`, `172.16.0.0/12`, or `192.168.0.0/16`.

Most PCF deployments do not require any modifications to this field.

To add your private network to the whitelist, perform the following steps:

1. Select **Advanced Features**.

2. Append a new `allow` rule to the existing contents of the **Whitelist for non-RFC-1918 Private Networks** field.


Include the word `allow`, the network CIDR range to allow, and a semi-colon (`;`) at the end. For example: `allow 172.99.0.0/24;`

3. Click **Save**.

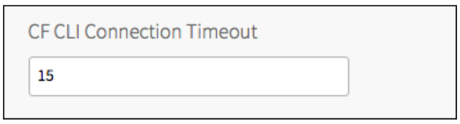
CF CLI Connection Timeout

The **CF CLI Connection Timeout** field allows you to override the default five second timeout of the Cloud Foundry Command Line Interface (cf CLI) used within your PCF deployment. This timeout affects the cf CLI command used to push PAS errand apps such as Notifications, Autoscaler, and Apps Manager.

Set the value of this field to a higher value, in seconds, if you are experiencing domain name resolution timeouts when pushing errands in PAS.

To modify the value of the **CF CLI Connection Timeout**, perform the following steps:

1. Select **Advanced Features**.

2. Add a value, in seconds, to the **CF CLI Connection Timeout** field.


3. Click **Save**.

Step 20: Configure Errands

Errands are scripts that Ops Manager runs automatically when it installs or uninstalls a product, such as a new version of Pivotal Application Service (PAS). There are two types of errands: *post-deploy errands* run after the product is installed, and *pre-delete errands* run before the product in uninstalled.

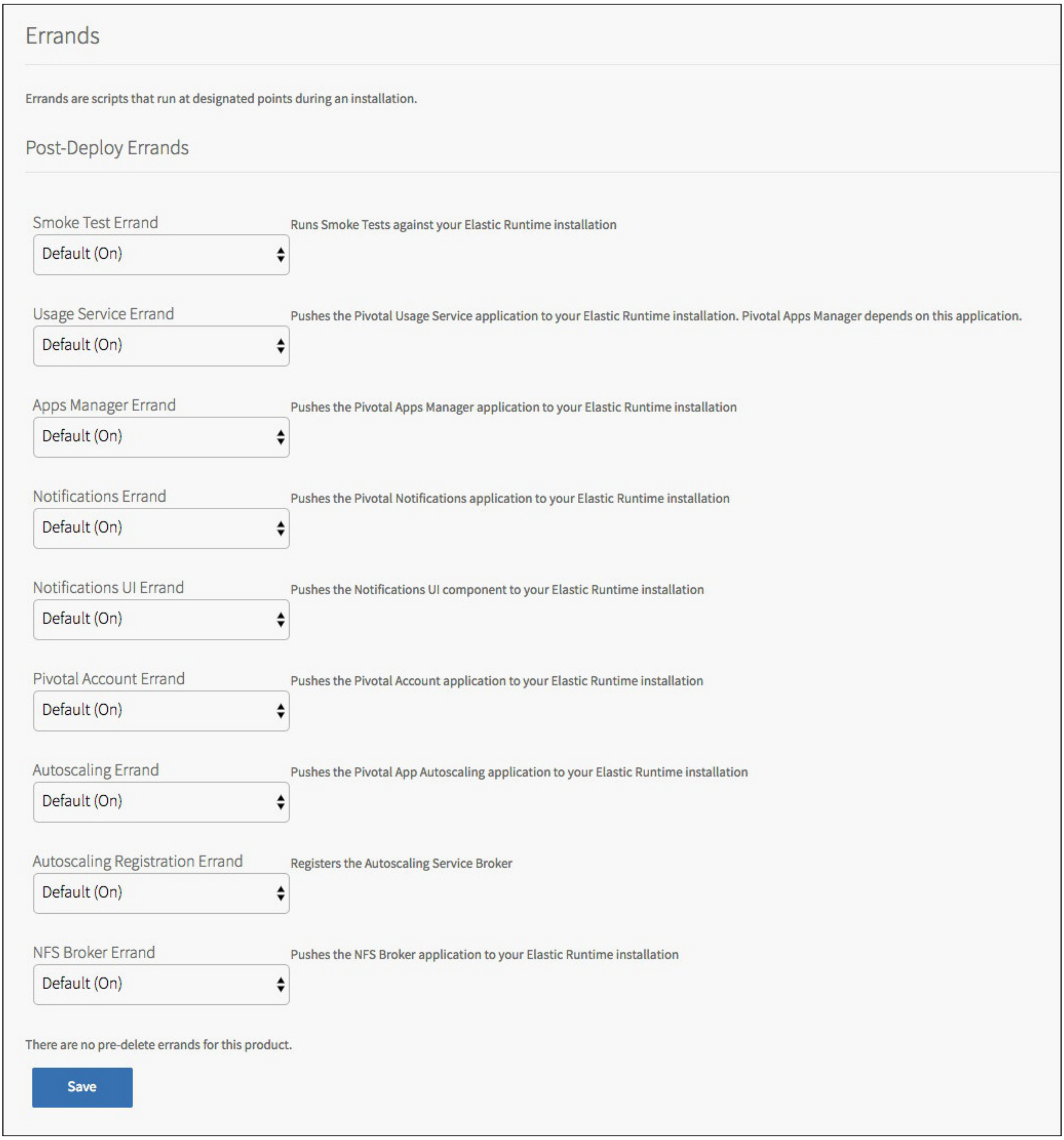
By default, Ops Manager always runs pre-delete errands, and only runs post-deploy errands when the product has changed since the last time Ops

Manager installed something. In PAS, the Smoke Test Errand defaults to always run.

The PAS tile **Errands** pane lets you change these run rules. For each errand, you can select **On** to run it always, **Off** to never run it, or **When Changed** to run it only when the product has changed since the last install.

For more information about how Ops Manager manages errands, see the Managing Errands in Ops Manager topic.

> **Note**: Several errands deploy apps that provide services for your deployment, such as Autoscaling and Notifications. Once one of these apps is running, selecting **Off** for the corresponding errand on a subsequent installation does not stop the app.



- **Smoke Test Errand** verifies that your deployment can do the following:
  - Push, scale, and delete apps
  - Create and delete orgs and spaces
- **Usage Service Errand** deploys the Pivotal Usage Service application, which Apps Manager depends on.
- **Apps Manager Errand** deploys Apps Manager, a dashboard for managing apps, services, orgs, users, and spaces. Until you deploy Apps Manager, you

must perform these functions through the cf CLI. After Apps Manager has been deployed, Pivotal recommends deselecting the checkbox for this errand on subsequent PAS deployments. For more information about Apps Manager, see the Getting Started with the Apps Manager topic.

- **Notifications Errand** deploys an API for sending email notifications to your PCF platform users.

> **Note**: The Notifications app requires that you configure SMTP with a username and password, even if you set the value of **SMTP Authentication Mechanism** to `none`.

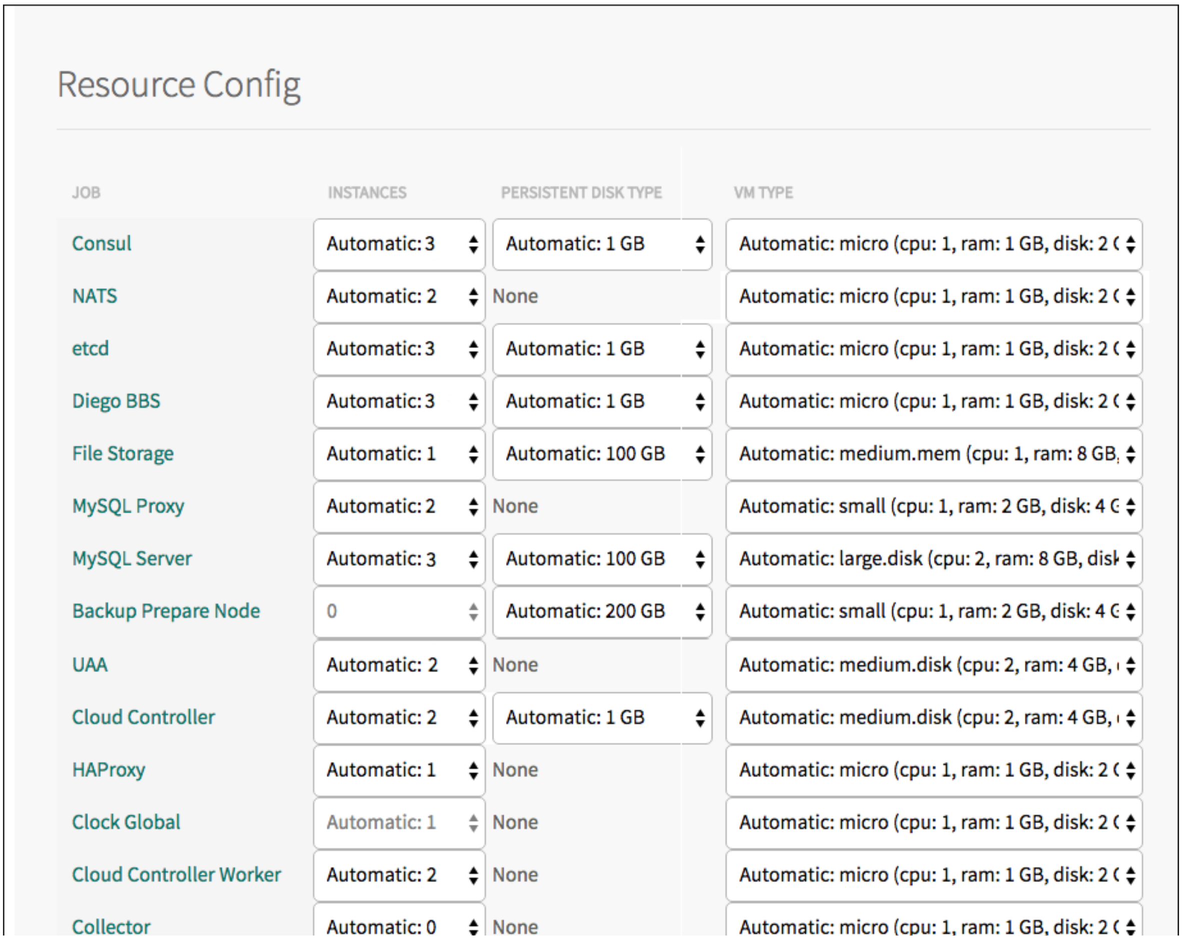
- **Notifications UI Errand** deploys a dashboard for users to manage notification subscriptions.
- **Pivotal Account Errand** deploys Pivotal Account, a dashboard that allows users to create and manage their accounts. In the Pivotal Account dashboard, users can launch applications, manage their profiles, manage account security, manage notifications, and manage approvals. See the Enabling Pivotal Account topic for more information.
- **Autoscaling Errand** enables you to configure your apps to automatically scale in response to changes in their usage load. See the Scaling an Application Using Autoscaler topic for more information.
- **Autoscaling Registration Errand** makes the Autoscaling service available to your applications. Without this errand, you cannot bind the Autoscaling app to your apps.
- **NFS Broker Errand** enables you to use NFS Volume Services by installing the NFS Broker app in PAS. See the Enabling NFS Volume Services topic for more information.

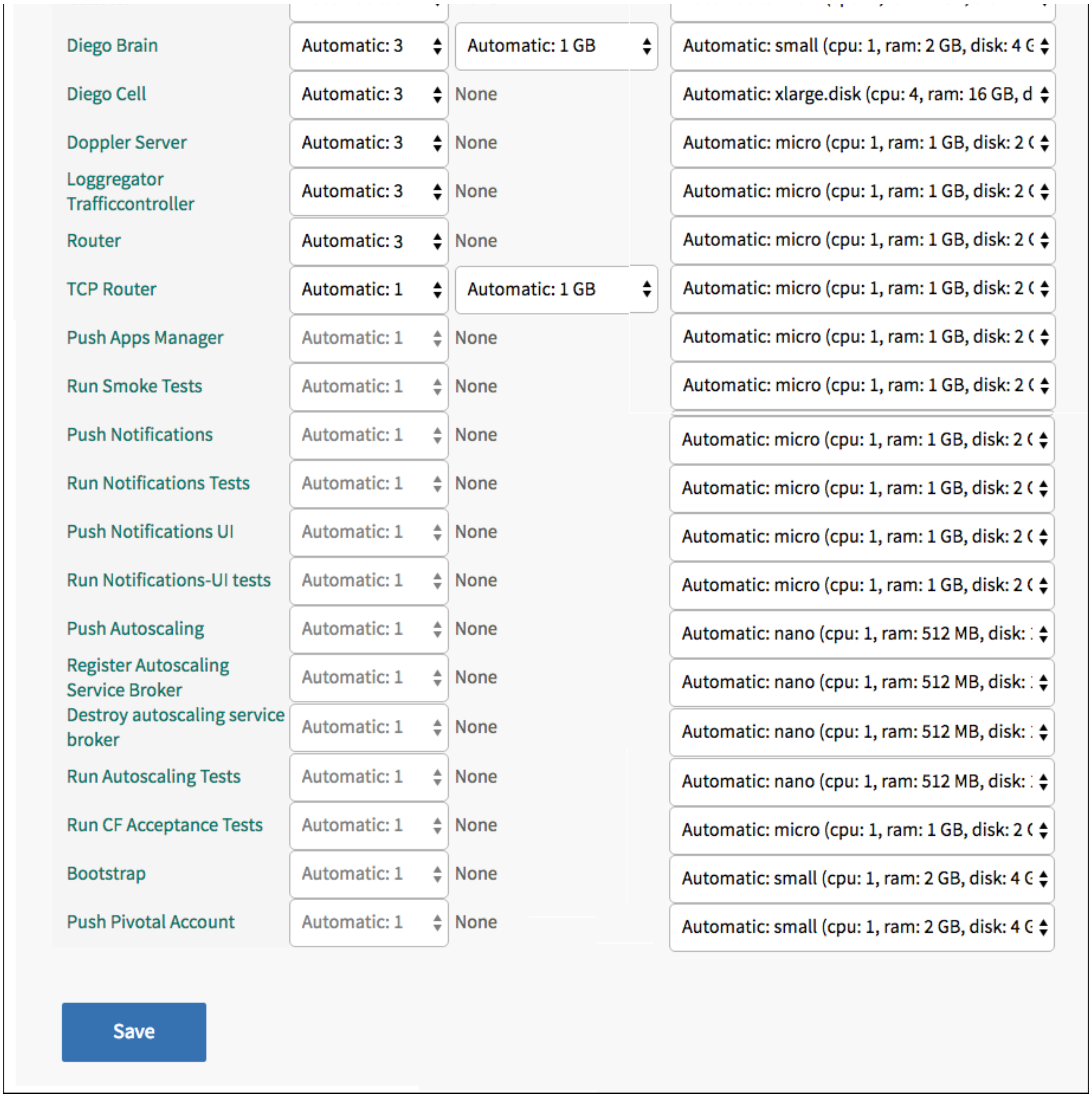
Step 21: (Optional) Configure Resources

> **Note**: Ops Manager 2.0 defines specific instance types with preset sizes for CPU, memory, and disk space. Ops Manager 1.6 and earlier required custom sizes for these three resources. With the upgrade from 1.6 to 1.7, each instance adopts the type that most closely matches its previous sizes. To change these resource allocations, select a different instance `type` under **Resource Config**.

PAS defaults to a highly available resource configuration. However, you may still need to perform additional procedures to make your deployment highly available. See the Zero Downtime Deployment and Scaling in CF and the Scaling Instances in PAS topics for more information.

If you do not want a highly available resource configuration, you must scale down your instances manually by navigating to the **Resource Config** section and using the drop-down menus under **Instances** for each job.





Diego Brain	Automatic: 3	Automatic: 1 GB	Automatic: small (cpu: 1, ram: 2 GB, disk: 4 G
Diego Cell	Automatic: 3	None	Automatic: xlarge.disk (cpu: 4, ram: 16 GB, d
Doppler Server	Automatic: 3	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 (
Loggregator Trafficcontroller	Automatic: 3	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 (
Router	Automatic: 3	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 (
TCP Router	Automatic: 1	Automatic: 1 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 (
Push Apps Manager	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 (
Run Smoke Tests	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 (
Push Notifications	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 (
Run Notifications Tests	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 (
Push Notifications UI	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 (
Run Notifications-UI tests	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 (
Push Autoscaling	Automatic: 1	None	Automatic: nano (cpu: 1, ram: 512 MB, disk: :
Register Autoscaling Service Broker	Automatic: 1	None	Automatic: nano (cpu: 1, ram: 512 MB, disk: :
Destroy autoscaling service broker	Automatic: 1	None	Automatic: nano (cpu: 1, ram: 512 MB, disk: :
Run Autoscaling Tests	Automatic: 1	None	Automatic: nano (cpu: 1, ram: 512 MB, disk: :
Run CF Acceptance Tests	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 (
Bootstrap	Automatic: 1	None	Automatic: small (cpu: 1, ram: 2 GB, disk: 4 G
Push Pivotal Account	Automatic: 1	None	Automatic: small (cpu: 1, ram: 2 GB, disk: 4 G

Save

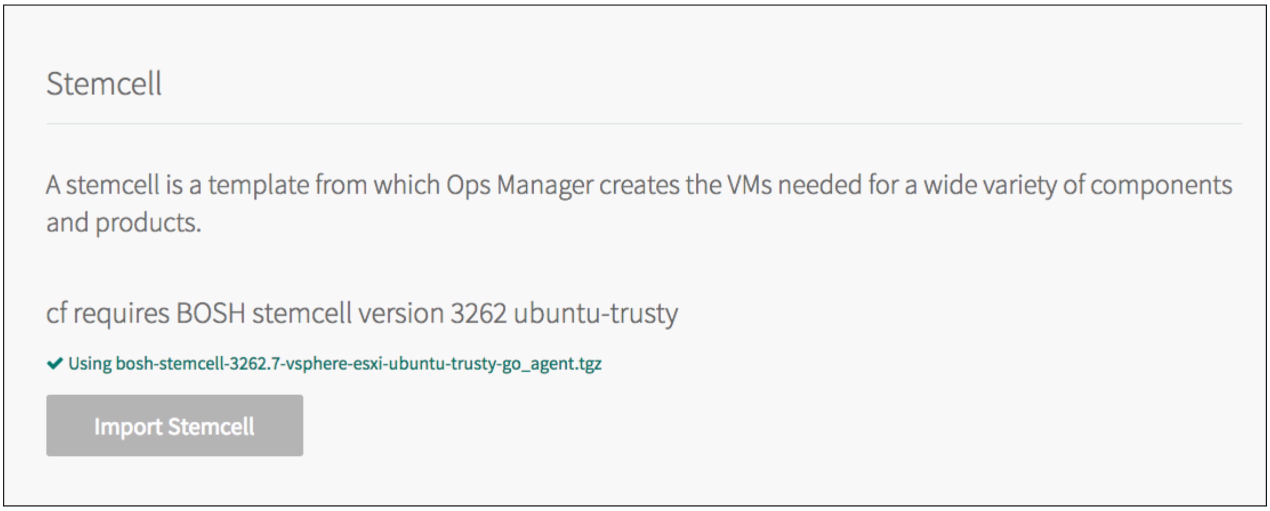
By default, PAS also uses an internal filestore and internal databases. If you configure PAS to use external resources, you can disable the corresponding system-provided resources in Ops Manager to reduce costs and administrative overhead.

Complete the following procedures to disable specific VMs in Ops Manager:

1. Click **Resource Config**.
2. If you configure PAS to use an external S3-compatible filestore, edit the following fields:
   - **File Storage**: Enter `0` in **Instances**.
3. If you selected **External** when configuring the UAA or System databases, edit the following fields:
   - **MySQL Proxy**: Enter `0` in **Instances**.
   - **MySQL Server**: Enter `0` in **Instances**.
   - **MySQL Monitor**: Enter `0` in **Instances**.
   - **Cloud Controller Database**: Enter `0` in **Instances**.
   - **UAA Database**: Enter `0` in **Instances**.
4. If you are using an External Load Balancer instead of HAProxy, enter `0` in the **Instances** field for **HAProxy**.
5. Click **Save**.

Step 22: Configure Stemcell

1. Select **Stemcell**. This page displays the stemcell version that shipped with Ops Manager.

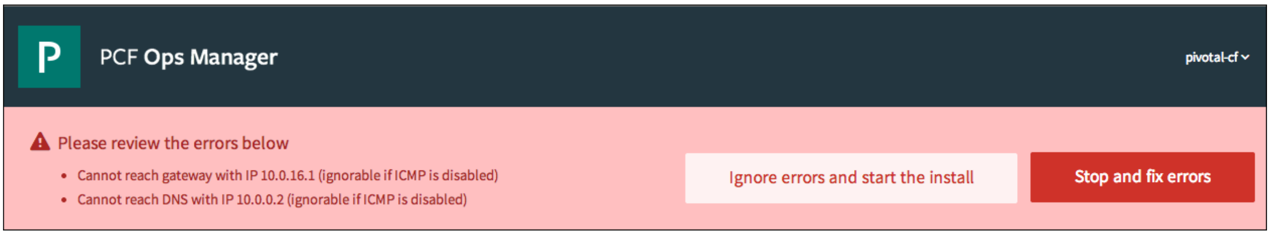


You can also use this page to import a new stemcell version. You only need to import a new Stemcell if your Ops Manager does not already have the Stemcell version required by PAS.

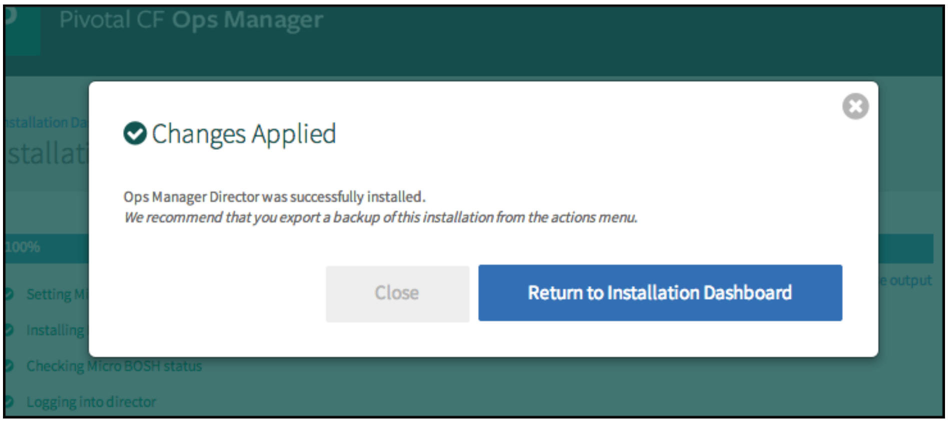
Step 23: Complete the PAS Installation

1. Click the **Installation Dashboard** link to return to the Installation Dashboard.

2. Click **Apply Changes**. If the following ICMP error message appears, click **Ignore errors and start the install**.



The install process generally requires a minimum of 90 minutes to complete. The image shows the Changes Applied window that displays when the installation process successfully completes.



vSphere Virtual Disk Types

Page last updated:

When you create a virtual machine in VMware vSphere, vSphere creates a new virtual hard drive for that virtual machine. The virtual hard drive is contained in a virtual machine disk (VMDK). The disk format you choose for the new virtual hard drive can have a significant impact on performance.

You can choose one of three formats when creating a virtual hard drive:

- Thin Provisioned
- Thick Provisioned Lazy Zeroed
- Thick Provisioned Eager Zeroed

Thin Provisioned

Advantages:

- Fastest to provision
- Allows disk space to be overcommitted to VMs

Disadvantages:

- Slowest performance due to metadata allocation overhead and additional overhead during initial write operations
- Overcommitment of storage can lead to application disruption or downtime if resources are actually used
- Does not support clustering features

When vSphere creates a thin provisioned disk, it only writes a small amount of metadata to the datastore. It does not allocate or zero out any disk space. At write time, vSphere first updates the allocation metadata for the VMDK, then zeros out the block or blocks, then finally writes the data. Because of this overhead, thin provisioned VMDKs have the lowest performance of the three disk formats.

Thin provisioning allows you to overcommit disk spaces to VMs on a datastore. For example, you could put 10 VMs, each with a 50 GB VMDK attached to it, on a single 100 GB datastore, as long as the sum total of all data written by the VMs never exceeded 100 GB. Thin provisioning allows administrators to use space on datastores that would otherwise be unavailable if using thick provisioning, possibly reducing costs and administrative overhead.

Thick Provisioned Lazy Zeroed

Advantages:

- Faster to provision than Thick Provisioned Eager Zeroed
- Better performance than Thin Provisioned

Disadvantages:

- Slightly slower to provision than Thin Provisioned
- Slower performance than Thick Provisioned Eager Zero
- Does not support clustering features

When vSphere creates a thick provisioned lazy zeroed disk, it allocates the maximum size of the disk to the VMDK, but does nothing else. At the initial access to each block, vSphere first zeros out the block, then writes the data. Performance of a thick provisioned lazy zeroed disk is not as good a thick provisioned eager zero disk because of this added overhead.

Thick Provisioned Eager Zeroed

Advantages:

- Best performance
- Overwriting allocated disk space with zeros reduces possible security risks
- Supports clustering features such as Microsoft Cluster Server (MSCS) and VMware Fault Tolerance

Disadvantages:

- Longest time to provision

When vSphere creates a thick provisioned eager zeroed disk, it allocates the maximum size of the disk to the VMDK, then zeros out all of that space.

Example: If you create an 80 GB thick provisioned eager zeroed VMDK, vSphere allocates 80 GB and writes 80 GB of zeros.

By overwriting all data in the allocated space with zeros, thick provisioned eager zeroed eliminates the possibility of reading any residual data from the disk, thereby reducing possible security risks.

Thick provisioned eager zeroed VMDKs have the best performance. When a write operation occurs to a thick provisioned eager zeroed disk, vSphere writes to the disk, with none of the additional overhead required by thin provisioned or thick provisioned lazy zeroed formats.

Using the Cisco Nexus 1000v Switch with Ops Manager

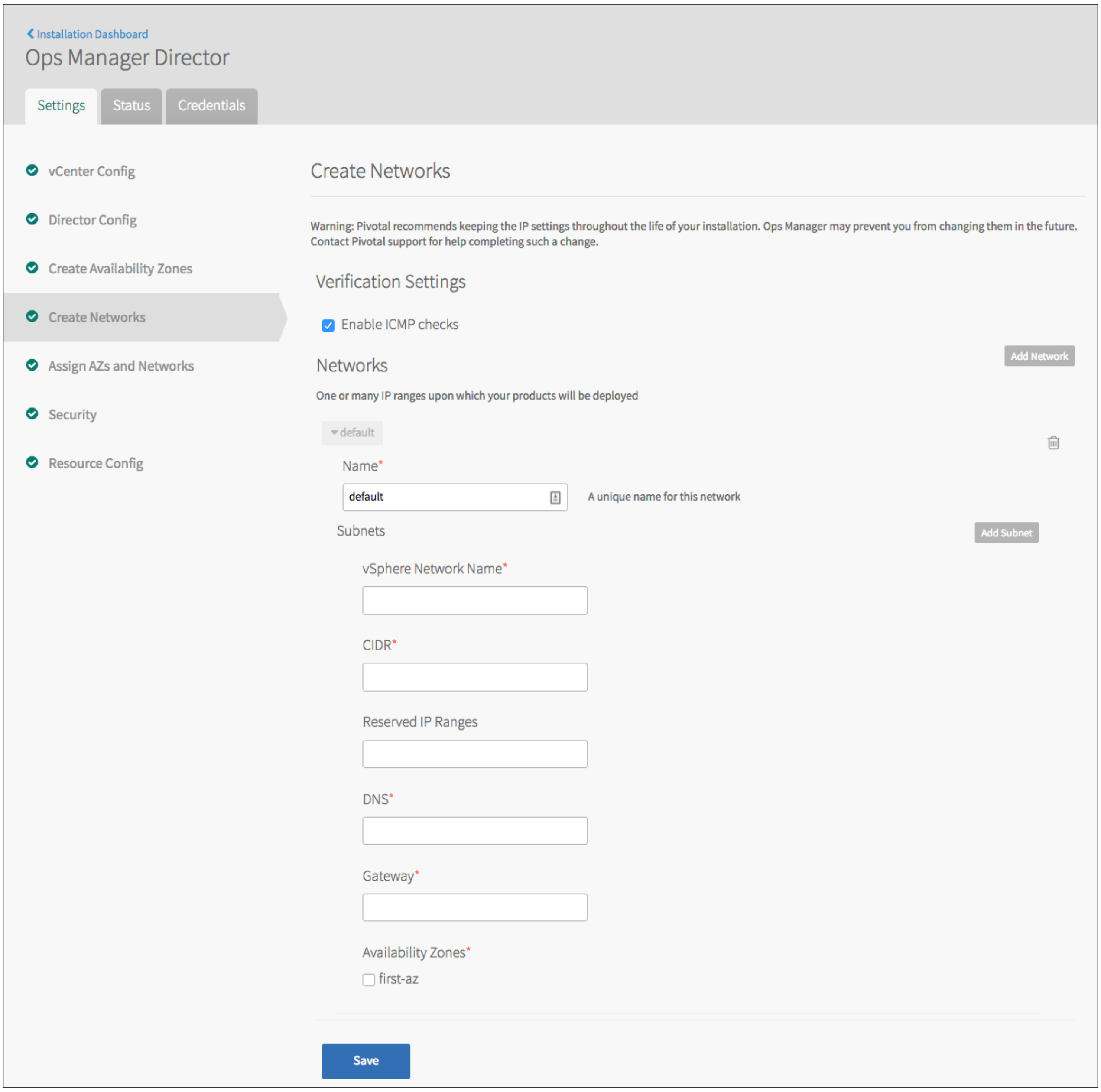
Page last updated:

Refer to the procedure in this topic to use Ops Manager with the Cisco Nexus 1000v Switch. First, configure Ops Manager through Step 4 in Configuring Ops Manager Director for VMware vSphere. Then configure your network according to the following steps.

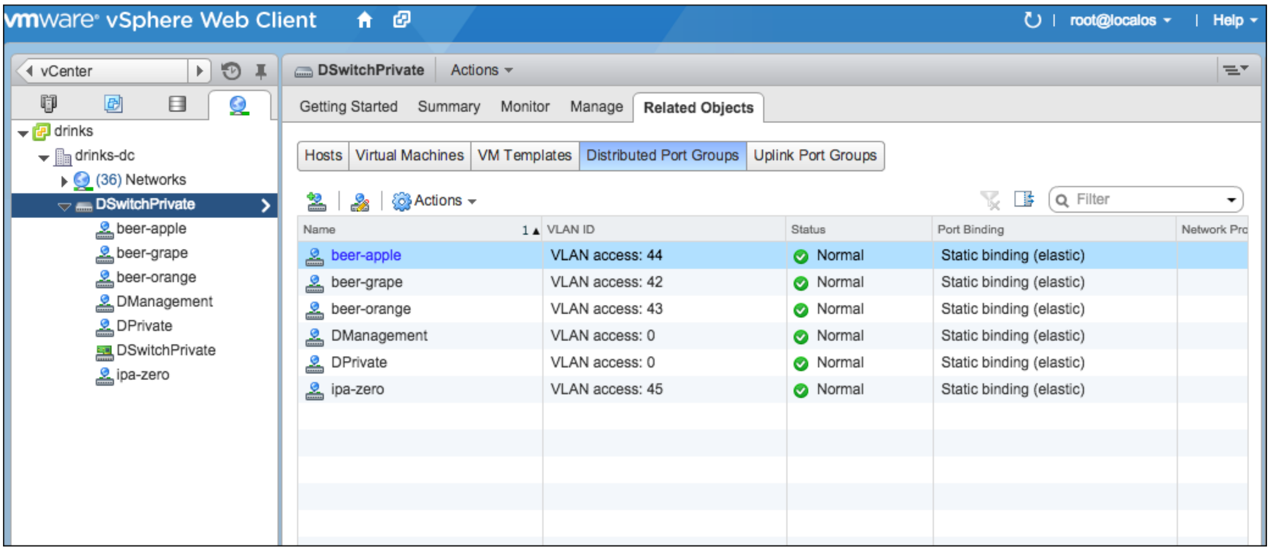
1. From your Pivotal Cloud Foundry (PCF) Ops Manager **Installation Dashboard**, click the **Ops Manager Director** tile.

2. Select **Create Networks**.

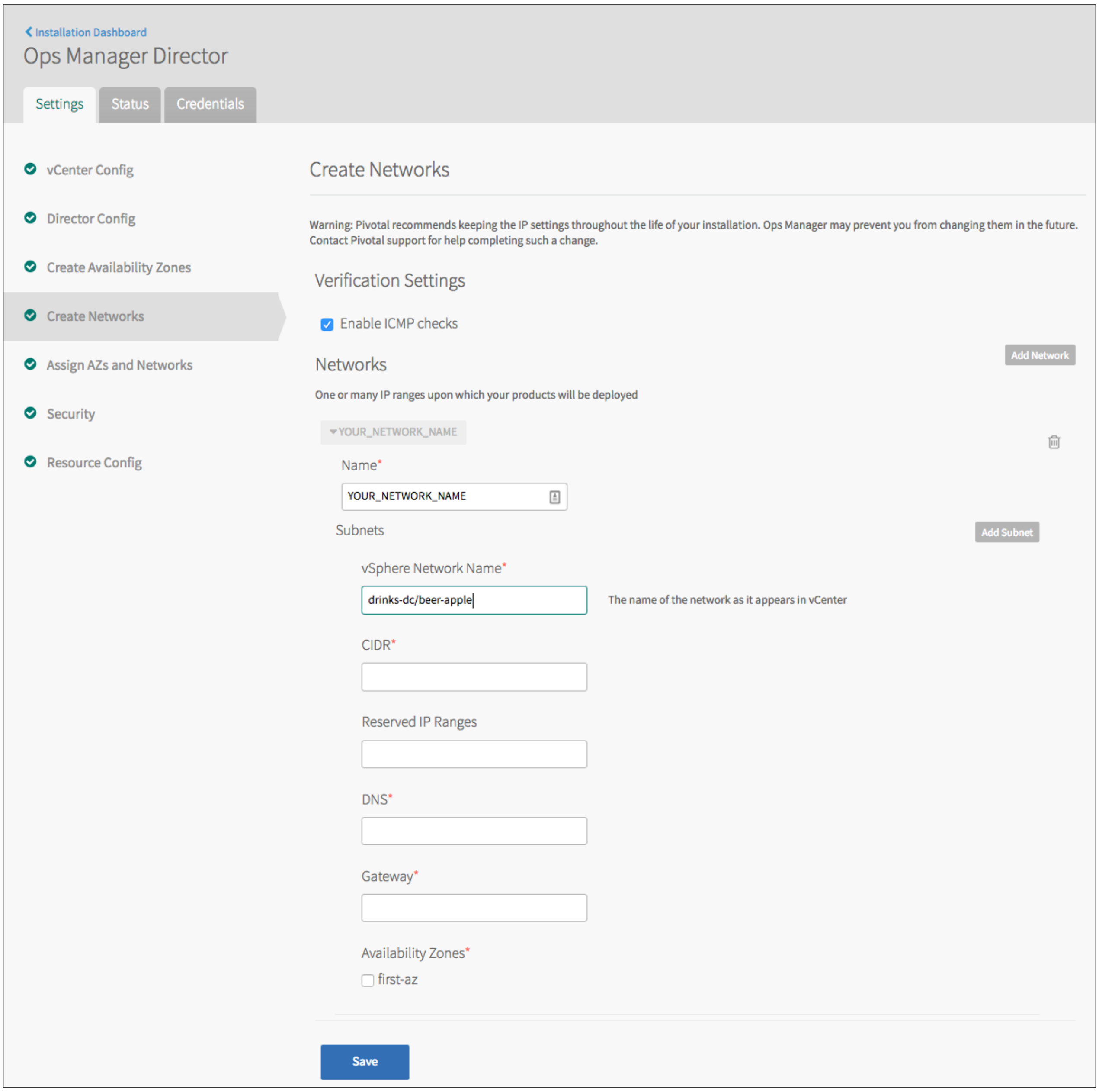
3. Click the network name to configure the network settings. This is **default** if you have not changed the name.



4. Find the folder name and port group name for the switch, as you configured them in vCenter. For the example vSphere environment pictured below, a user might want to use the switch configured on the `beer-apple` port group, which is in the `drinks-dc` folder.



5. In the **vSphere Network Name** field, instead of entering your network name, enter the folder name and port group name for the switch, as you configured them in vCenter. For the example vSphere environment pictured above, you would enter `drinks-dc/beer-apple` to use the switch configured on the `beer-apple` port group.



6. Click **Save**.

7. Return to [Configuring Ops Manager Director for VMware vSphere] to complete the Ops Manager installation.

Using Ops Manager Resurrector on vSphere

Page last updated:

The Ops Manager Resurrector increases Pivotal Application Service (PAS) availability in the following ways:

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

The Ops Manager Resurrector continuously monitors the status of all virtual machines in an PAS deployment. The Resurrector also monitors the BOSH Agent on each VM. If either the VM or the BOSH Agent fail, the Resurrector restarts the virtual machine on another active host.

Limitations

The following limitations apply to using the Ops Manager Resurrector:

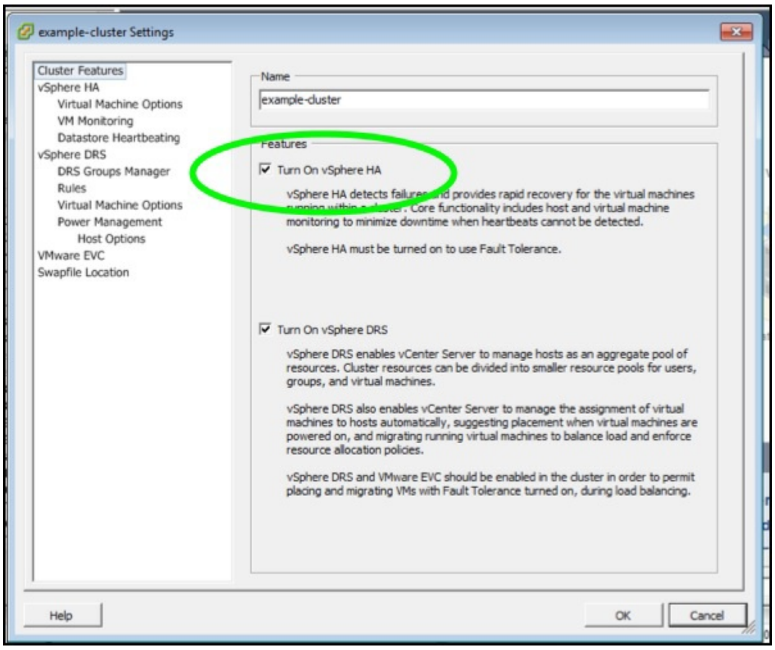
- The Resurrector does not monitor or protect the Ops Manager VM or the BOSH Director VM.
- The Resurrector might not be able to resolve issues caused by the loss of an entire host.
- The Resurrector does not monitor or protect data storage.

For increased reliability, in addition to using BOSH Resurrector, Pivotal recommends that you use vSphere High Availability to protect all of the VMs in your deployment, and that you use a highly-available storage solution.

Enabling vSphere High Availability

Follow the steps below to enable vSphere High Availability:

1. Launch the vSphere Management Console.
2. Right-click the cluster that contains the Pivotal Cloud Foundry (PCF) deployment and select **Edit Settings**.
3. Check the **Turn on vSphere High Availability** checkbox.



4. Click **OK** to enable vSphere High Availability on the cluster.

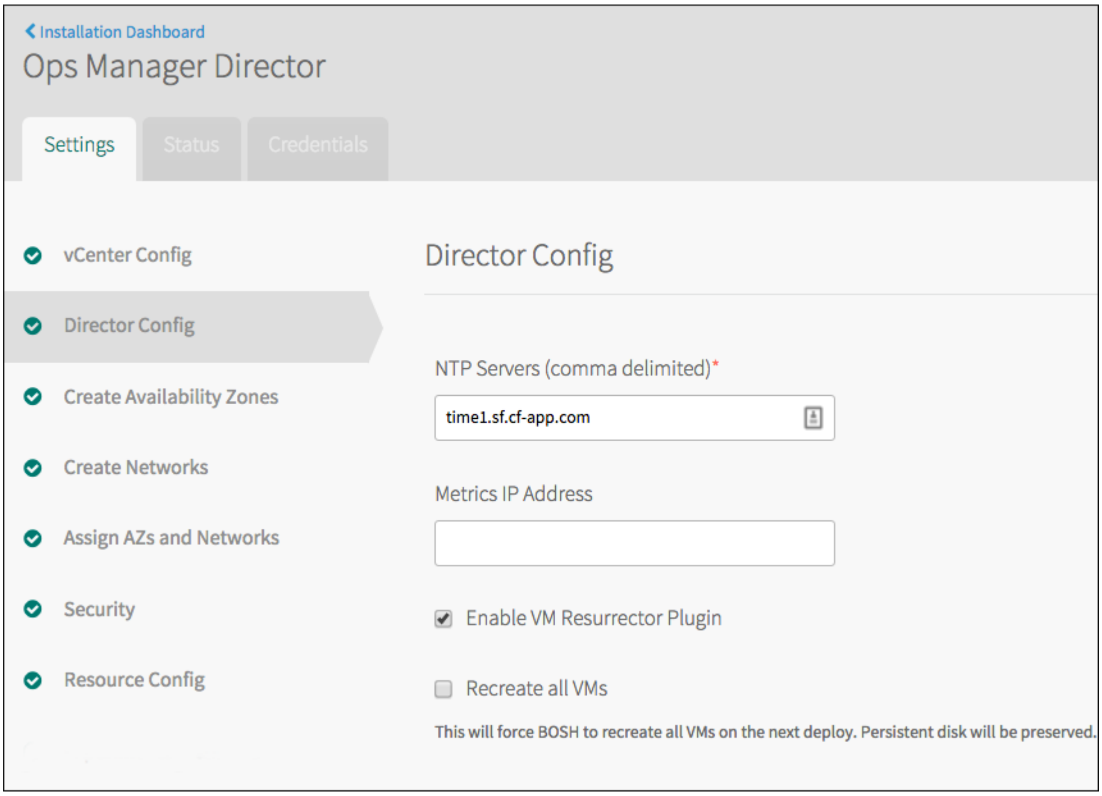
Enabling Ops Manager Resurrector

To enable the Ops Manager Resurrector:

1. Log into the Ops Manager web interface.

2. On the Product Dashboard, select **Ops Manager Director**.

3. In the left navigation menu, select **Director Config**.



4. Check **Enable VM Resurrector Plugin** and click **Save**.

Configuring Pivotal Cloud Foundry SSL Termination for vSphere Deployments

Page last updated:

To use SSL termination in Pivotal Cloud Foundry (PCF), you must configure the Pivotal-deployed HAProxy load balancer or your own load balancer.

Pivotal recommends that you use HAProxy in lab and test environments only. Production environments should instead use a highly-available customer-provided load balancing solution.

Select an SSL termination method to determine the steps you must take to configure Pivotal Application Service (PAS).

Using the Pivotal HAProxy Load Balancer

PCF deploys with a single instance of HAProxy for use in lab and test environments. You can use this HAProxy instance for SSL termination and load balancing to the PCF Routers. HAProxy can generate a self-signed certificate if you do not want to obtain a signed certificate from a well-known certificate authority.

Note: Certificates generated in PAS are signed by the Operations Manager Certificate Authority. They are not technically self-signed, but they are referred to as “Self-Signed Certificates” in the Ops Manager GUI and throughout this documentation.

To use the HAProxy load balancer, you must create a wildcard A record in your DNS and configure three fields in the PAS product tile.

1. Create an A record in your DNS that points to the HAProxy IP address. The A record associates the **System Domain** and **Apps Domain** that you configure in the **Domains** section of the PAS tile with the HAProxy IP address.

   For example, with `cf.example.com` as the main subdomain for your CF install and an HAProxy IP address `203.0.113.1`, you must create an A record in your DNS that serves `example.com` and points `*.cf` to `203.0.113.1`.

Name	Type	Data	Domain
*.cf	A	203.0.113.1	example.com

2. Use the Linux `host` command to test your DNS entry. The `host` command should return your HAProxy IP address.

   Example:

   ```
   $ host cf.example.com
   cf.example.com has address 203.0.113.1
   $ host anything.example.com
   anything.cf.example.com has address 203.0.113.1
   ```

3. From the PCF Ops Manager Dashboard, click on the PAS tile.

4. Select **Networking**.

5. Leave the **Router IPs** field blank. HAProxy assigns the router IPs internally.

6. Enter the IP address for HAProxy in the **HAProxy IPs** field.

7. Provide your SSL certificate in the **SSL Termination Certificate and Private Key** field. See Providing a Certificate for your SSL Termination Point for details.

Return to the Getting Started Guide

Using Another Load Balancer

Production environments should use a highly-available customer-provided load balancing solution that does the following:

- Provides SSL termination with wildcard DNS location
- Provides load balancing to each of the PCF Router IPs

- Adds appropriate `x-forwarded-for` and `x-forwarded-proto` HTTP headers

You must register static IP addresses for PCF with your load balancer and configure three fields in the PAS product tile.

1. Register one or more static IP address for PCF with your load balancer.

2. Create an A record in your DNS that points to your load balancer IP address. The A record associates the **System Domain** and **Apps Domain** that you configure in the **Domains** section of the PAS tile with the IP address of your load balancer.

   For example, with `cf.example.com` as the main subdomain for your CF install and a load balancer IP address `198.51.100.1`, you must create an A record in your DNS that serves `example.com` and points `*.cf` to `198.51.100.1`.

Name	Type	Data	Domain
*.cf	A	198.51.100.1	example.com

3. From the PCF Ops Manager Dashboard, click on the PAS tile.

4. Select **Networking**.

5. In the **Router IPs** field, enter the static IP address for PCF that you have registered with your load balancer.

6. Leave the **HAProxy IPs** field blank.

7. Provide your SSL certificate in the **SSL Termination Certificate and Private Key** field. See Providing a Certificate for your SSL Termination Point for details.

> **Note**: When adding or removing PCF routers, you must update your load balancing solution configuration with the appropriate IP addresses.

Return to the Installing Pivotal Cloud Foundry Guide

Availability Zones in vSphere

Page last updated:

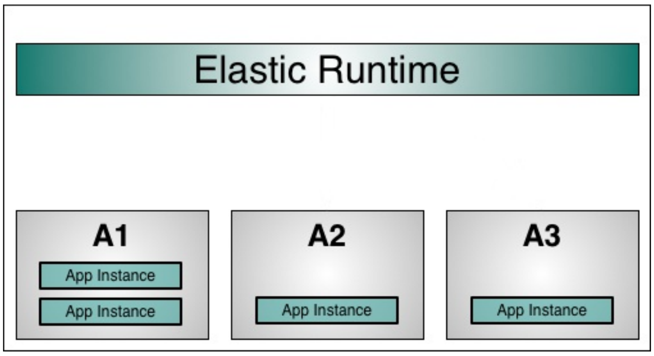
Pivotal defines an Availability Zone (AZ) as an operator-assigned, functionally independent segment of network infrastructure. In cases of partial infrastructure failure, Pivotal Application Service (PAS) distributes and balances all instances of running applications across remaining AZs. Strategic use of Availability Zones contributes to the fault tolerance and high availability of a PAS deployment.

PAS on VMware vSphere supports distributing deployments across multiple AZs. See the section on AZs in Configuring Ops Manager Director for VMware vSphere.

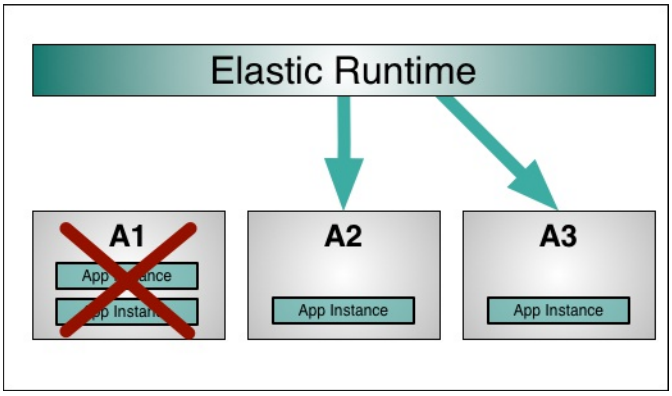
It is recommended that customers use three Availability Zones to operate a highly available installation of PAS.

Balancing Across AZs During Failure: Example Scenario

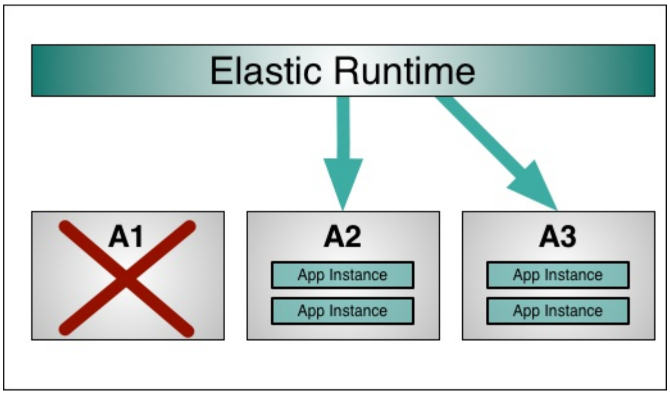
An operator scales an application to four instances in an PAS environment distributed across three availability zones: A1, A2, and A3. The environment allocates the instances according to the Diego Auction.



If A1 experiences a power outage or hardware failure, the two application instances running in A1 terminate while the application instances in zones A2 and A3 continue to run:



If A1 remains unavailable, PAS balances new instances of the application across the remaining availability zones:



Updating NSX Security Group and Load Balancer Information

Page last updated:

This topic describes how to update security group and load balancer information for Pivotal Cloud Foundry (PCF) deployments using NSX-V on vSphere. To update this information, you must use the Ops Manager API.

See the Ops Manager API documentation at `https://YOUR-OPS-MANAGER-FQDN/docs` for more information about the API.

Note: Ops Manager v1.11 supports NSX-V 6.2+.

Authenticate

To use the Ops Manager API, you must authenticate and retrieve a token from the Ops Manager User Account and Authentication (UAA) server. For instructions, see the Using Ops Manager API topic.

Update Security Group or Load Balancer Information

To update either NSX security group or load balancer information, you use `curl` to make a PUT request against the `api/v0/staged/products/product_guid/jobs/job_guid/resource_config` endpoint.

You must first retrieve the GUID of your PCF deployment, and the GUID of the job whose information you want to update.

Perform the following steps:

1. Retrieve a list of staged products:

```
$ curl 'https://OPS-MAN-FQDN/api/v0/staged/products' \
-H "Authorization: Bearer UAA-ACCESS-TOKEN"
[
  {
    "product_version" : "1.10.6.0",
    "guid" : "p-bosh-dee11e111e1111ee1e1a",
    "installation_name" : "p-bosh",
    "type" : "p-bosh"
  },
  {
    "type" : "cf",
    "product_version" : "1.10.8-build.7",
    "installation_name" : "cf-01222ab1111111aaa1a",
    "guid" : "cf-01222ab1111111aaa1a"
  }
]
```

   Record the GUID of the `cf` product. In the above example, the GUID is `cf-01222ab1111111aaa1a`.

2. Retrieve a list of jobs for your product:

```
$ curl 'https://OPS-MAN-FQDN/api/v0/staged/products/PRODUCT-GUID/jobs' \
-H "Authorization: Bearer UAA-ACCESS-TOKEN'
{
"jobs" : [
  {
    "guid" : "consul_server-9c37cf48ae7412f2afd1",
    "name" : "consul_server"
  },
  {
    "name" : "nats",
    "guid" : "nats-6af18efdd18d198edee9"
  },
  {
    "name" : "nfs_server",
    "guid" : "nfs_server-b49b0b2aed247302c0e1"
  },
  ...
```

   Record the GUID of the job whose security groups you want to update.

3. You can update either your security group information, load balancer information, or both.
    - **Security groups**: To update the security groups for your job, use the following command:

    ```
    $ curl "https://OPS-MAN-FQDN/api/v0/staged/products/PRODUCT-GUID/jobs/JOB-GUID/resource_config" \
      -X PUT \
      -H "Authorization: Bearer UAA-ACCESS-TOKEN" \
      -d '{"nsx_security_groups": ["SECURITY-GROUP1", "SECURITY-GROUP2"]}'
    ```

    The value for `nsx_security_groups` is a list of the security groups you want to set for the job. To clear all security groups for a job, pass an empty list with the value `[]`.

    - **Load balancers**: To update the load balancers for your job, use the following command:

    ```
    $ curl "https://OPS-MAN-FQDN/api/v0/staged/products/PRODUCT-GUID/jobs/JOB-GUID/resource_config" \
      -X PUT \
      -H "Authorization: Bearer UAA-ACCESS-TOKEN" \
      -d '{"nsx_lbs": \
           { \
            "edge_name": "EDGE-NAME", \
            "pool_name": "POOL-NAME", \
            "security_group": "SECURITY-GROUP", \
            "port": "PORT-NUMBER" \
           }
    ```

    Replace the placeholder values under `nsx_lbs` as follows:

        - `EDGE-NAME`: The name of the NSX Edge
        - `POOL-NAME`: The name of the NSX Edge's server pool
        - `SECURITY-GROUP`: The name of the NSX server pool's target security group
        - `PORT`: The name of the port that the VM service is listening on, such as `5000`

4. Navigate to `OPS-MAN-FQDN` in a browser and log in to the Ops Manager Installation Dashboard.

5. Click **Apply Changes** to redeploy.

Installing PCF Isolation Segment

Page last updated:

This topic describes how to install the PCF Isolation Segment tile, which allows operators to isolate deployment workloads into dedicated resource pools called *isolation segments*.

Installing the tile installs a single isolation segment. However, you can install multiple isolation segments using the Replicator tool documented in Step 4.

After installing the tile, you must perform the steps in the Create an Isolation Segment section of the *Managing Isolation Segments* topic to create the isolation segment in the Cloud Controller Database (CCDB). The topic also includes information about managing an isolation segment.

For more information about how isolation segments work, see the Isolation Segments section of the *Understanding Cloud Foundry Security* topic.

Step 1: (Optional) Configure Routing

By default, the Pivotal Application Service (PAS) Router handles traffic for your isolation segment. However, you can deploy a dedicated router for your isolation segment instead. For information about configuring and managing routing for isolation segments, see the Routing for Isolation Segments topic.

To deploy a dedicated router, perform the following steps:

1. Add a load balancer in front of the PAS Router. The steps to do this depend on your IaaS, but the setup of the load balancer should mirror the setup of the load balancer for the PAS Router that you configured in the PAS tile.
2. Create a wildcard DNS entry for traffic routed to any app in the isolation segment. For example, `*.iso.example.com`.
3. Attach the wildcard DNS entry to the load balancer you created.

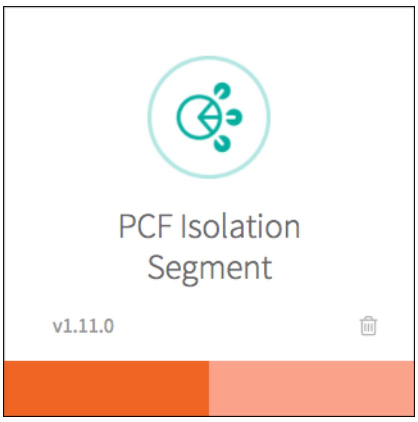
Step 2: Install the Tile

Perform the following steps to install the PCF Isolation Segment tile:

1. Download the product file from Pivotal Network.
2. Navigate to `YOUR-OPSMAN-FQDN` in a browser to log in to the Ops Manager Installation Dashboard.
3. Click **Import a Product** and select the downloaded product file.
4. Under **PCF Isolation Segment** in the left column, click the plus sign.

Step 3: Configure the Tile

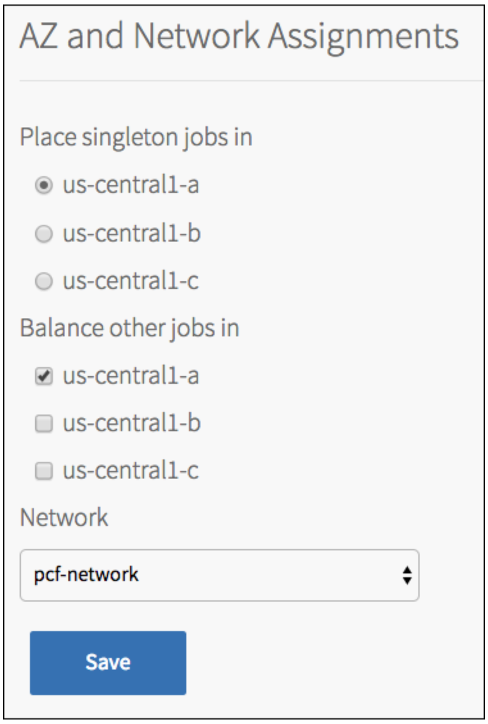
Click the orange **PCF Isolation Segment** tile to start the configuration process.



Assign AZs and Networks

Perform the following steps to configure the PCF Isolation Segment tile:

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



2. Select an availability zone for your singleton jobs, and one or more availability zones to balance other jobs in.

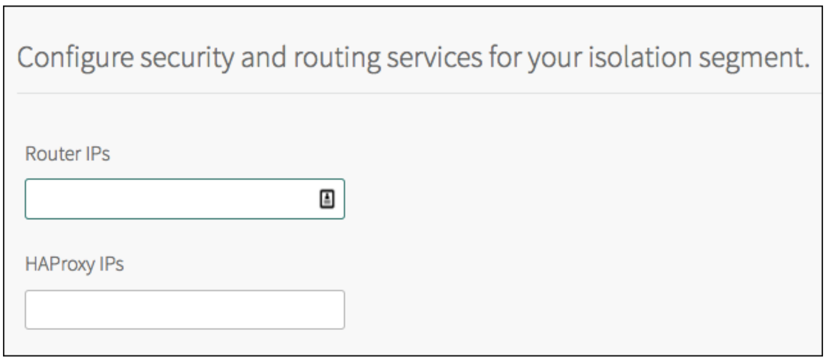
3. Select a network. This network does not need to be the same network where you deployed PAS. For most deployments, operators should create unique networks in which to deploy the Isolation Segment tile. These networks should maintain network reachability with the Diego components so that the cells can reach the Diego Brain and Diego Database VMs.

4. Click **Save**.

Networking

Perform the following steps to configure the PCF Isolation Segment tile:

1. Click **Networking**.



2. (Optional): Under **Router IPs**, enter one or more static IP addresses for the routers that handle this isolation segment. These IP addresses must be within the subnet CIDR block that you defined in the Ops Manager network configuration for your Isolation Segment. If you have a load balancer, configure it to point to these IP addresses.

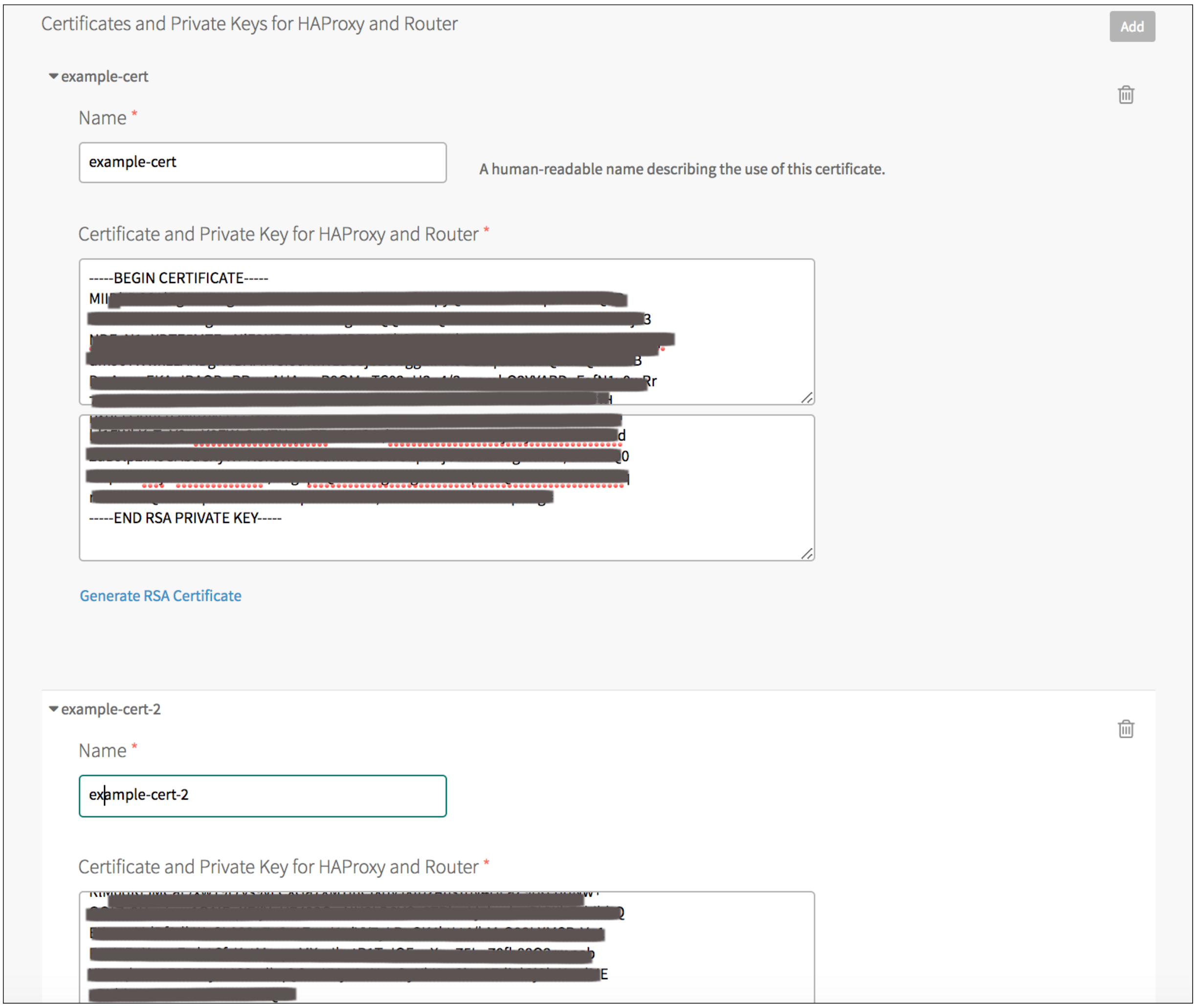
   > **Note**: Entering the static IP addresses is not necessary for deployments running on a public IaaS such as AWS, GCP, or Azure because PCF users specify the IaaS load balancer in the **Resource Config** section of the **PCF Isolation Segment** tile.

3. If you want to use HAProxy for this isolation segment, enter at least one address in the **HAProxy IPs** field. You should specify more than one IP address for high availability. Then configure your load balancer to forward requests for the domains you have set up for your deployment to these IP addresses. For more information, see Configuring SSL/TLS Termination at HAProxy.

> **Note**: If you rely on HAProxy for a feature in Pivotal Application Service (PAS) and you want isolated networking for this isolation segment, you may want to deploy the HAProxy provided by the Isolation Segment tile.

4. Under **Certificates and Private Key for HAProxy and Router**, you must provide at least one **Certificate and Private Key** name and certificate keypair for HAProxy and Gorouter. The HAProxy and Gorouter are enabled to receive TLS communication by default. You can configure multiple certificates for HAProxy and Gorouter.

    a. Click the **Add** button to add a name for the certificate chain and its private keypair. This certificate is the default used by Gorouter and HAProxy.

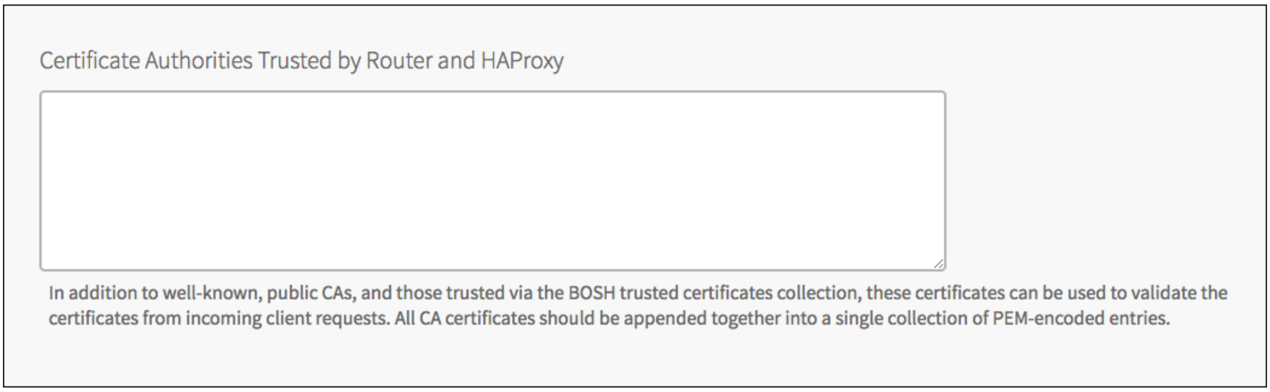
    

    You can either provide a certificate signed by a Certificate Authority (CA) or click on the **Generate RSA Certificate** link to generate a self-signed certificate in Ops Manager.

    b. If you want to configure multiple certificates for HAProxy and Gorouter, click the **Add** button and fill in the appropriate fields for each additional certificate keypair.

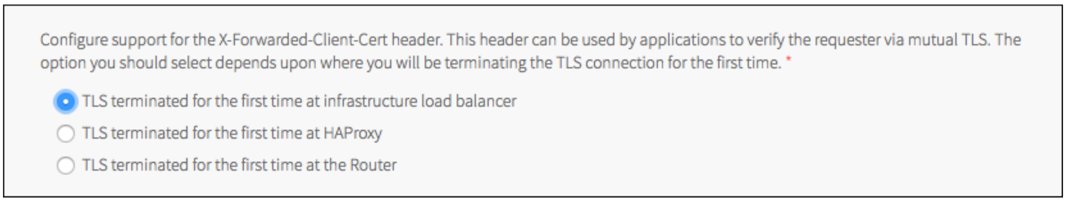
    For details about generating certificates in Ops Manager for your wildcard system domains, see the Providing a Certificate for Your SSL/TLS Termination Point topic.

5. (Optional) When validating client requests using mutual TLS, the Gorouter trusts multiple certificate authorities (CAs) by default. If you want to configure the Gorouter and HAProxy to trust additional CAs, enter your CA certificates under **Certificate Authorities Trusted by Router and HAProxy**. All CA certificates should be appended together into a single collection of PEM-encoded entries.



6. In the **Minimum version of TLS supported by HAProxy and Router** field, select the minimum version of TLS to use in HAProxy and Router communications. HAProxy and Router use TLS v1.2 by default. If you need to accommodate clients that use an older version of TLS, select a lower minimum version. For a list of TLS ciphers supported by the Gorouter, see Securing Traffic into Cloud Foundry.

7. Under **Configure support for the X-Forwarded-Client-Cert header**, configure PCF handles `x-forwarded-client-cert` (XFCC) HTTP headers based on where TLS is terminated for the first time in your deployment.



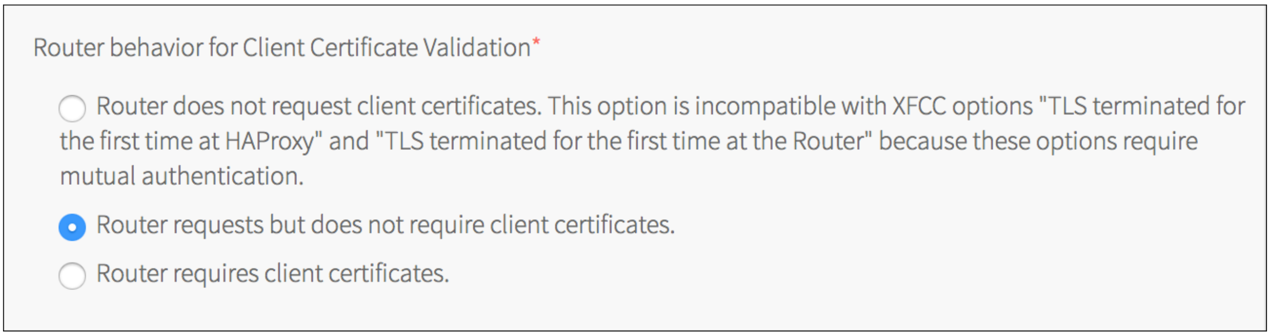
The following table indicates which option to choose based on your deployment layout.

If your deployment is configured as follows:	Then select the following option:	Additional notes:
◦ The Load Balancer is terminating TLS, and ◦ Load balancer is configured to put the client certificate from a mutual authentication TLS handshake into the X-Forwarded-Client-Cert HTTP header	TLS terminated for the first time at infrastructure load balancer (default).	Both HAProxy and the Gorouter forward the XFCC header when included in the request.
◦ The Load Balancer is configured to pass through the TLS handshake via TCP to the instances of HAProxy, and ◦ HAProxy instance count is > 0	TLS terminated for the first time at HAProxy.	HAProxy sets the XFCC header with the client certificate received in the TLS handshake. The Gorouter forwards the header. **Breaking Change**: In the **Router behavior for Client Certificates** field, you cannot select the **Router does not request client certificates** option.
◦ The Load Balancer is configured to pass through the TLS handshake via TCP to instances of the Gorouter	TLS terminated for the first time at the Gorouter.	The Gorouter strips the XFCC header if it is included in the request and forwards the client certificate received in the TLS handshake in a new XFCC header. If you have deployed instances of HAProxy, app traffic bypasses those instances in this configuration. If you have also configured your load balancer to route requests for ssh directly to the Diego Brain, consider reducing HAProxy instances to 0.

		Breaking Change: In the **Router behavior for Client Certificates** field, you cannot select the **Router does not request client certificates** option.

For a description of the behavior of each configuration option, see Forward Client Certificate to Applications.

8. To configure Gorouter behavior for handling client certificates, select one of the following options in the **Router behavior for Client Certificate Validation** field.



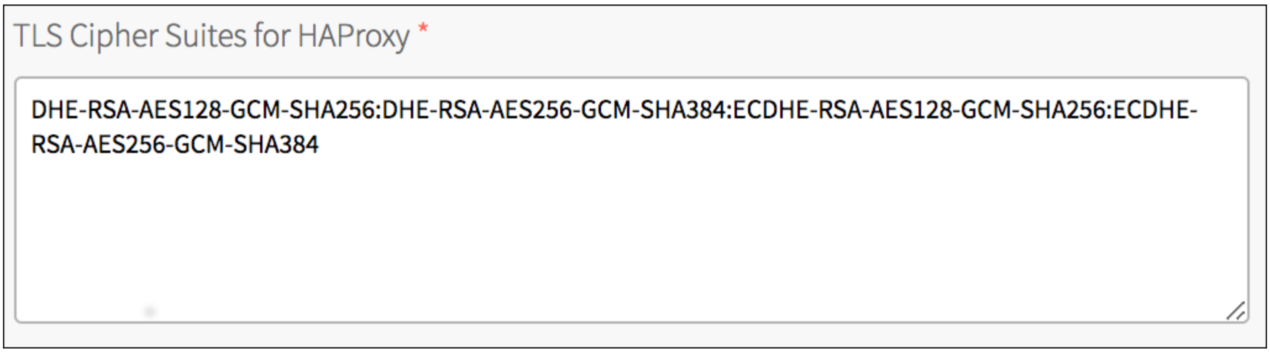
   - **Router does not request client certificates.** This option is incompatible with the XFCC configuration options **TLS terminated for the first time at HAProxy** and **TLS terminated for the first time at the Router** in PAS because these options require mutual authentication. As client certificates are not requested, client will not provide them, and thus validation of client certificates will not occur.
   - **Router requests but does not require client certificates.** Gorouter requests client certificates in TLS handshakes, validates them when presented, but does not require them. This is the default configuration.
   - **Router requires client certificates.** Gorouter validates that the client certificate is signed by a Certificate Authority that Gorouter trusts. If Gorouter cannot validate the client certificate, the TLS handshake fails.

9. In the **TLS Cipher Suites for Router** field, specify the TLS cipher suites to use for TLS handshakes between the Gorouter and downstream clients like load balancers or HAProxy. Use an ordered, colon-delimited list of Golang-supported TLS cipher suites in the OpenSSL format. The recommended setting is `ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384`. Operators should verify that the ciphers are supported by any clients or downstream components that will initiate TLS handshakes with the Gorouter. For a list of TLS ciphers supported by the Gorouter, see Securing Traffic into Cloud Foundry.

   Verify that whatever client is participating in the TLS handshake with the Gorouter has at least one cipher suite in common with the Gorouter.

   > **Note**: Specify cipher suites that are supported by the versions configured in the **Minimum version of TLS supported by HAProxy and Router** field.

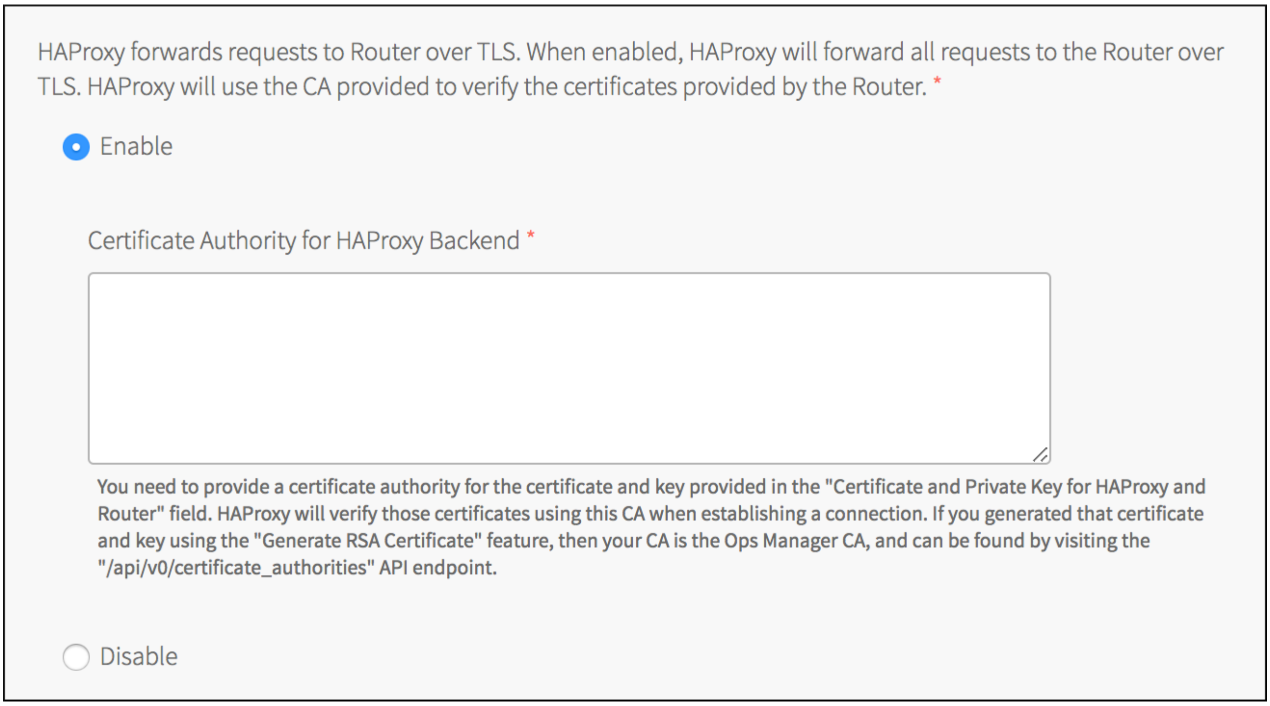
10. In the **TLS Cipher Suites for HAProxy** field, specify the TLS cipher suites to use in HAProxy for TLS handshakes between HAProxy and its clients such as load balancers and Gorouter. Use an ordered, colon-delimited list of TLS cipher suites in the OpenSSL format. The recommended setting: `DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384`
    Operators should verify that the ciphers are supported by any clients or downstream components that will initiate TLS handshakes with the HAProxy.



Verify that whatever clients are participating in the TLS handshake with HAProxy have at least one cipher suite in common with HAProxy.

> **Note:** Specify cipher suites that are supported by the versions configured in the **Minimum version of TLS supported by HAProxy and Router** field.

11. Under **HAProxy forwards requests to Router over TLS**, select **Enable** or **Disable** based on your deployment layout.



- **Enable HAProxy forwarding of requests to Router over TLS**

If you want to:	Encrypt communication between HAProxy and the Gorouter
Then configure the following:	1. Leave **Enable** selected. 2. In the **Certificate Authority for HAProxy Backend** field, specify the Certificate Authority (CA) that signed the certificate you configured in the **Certificate and Private Key for HAProxy and Router** field. **Note:** If you used the **Generate RSA Certificate** link to generate a self-signed certificate, then the CA to specify is the Ops Manager CA, which you can locate at the CA endpoint in the Ops Manager API. 3. Make sure that Gorouter and HAProxy have TLS cipher suites in common in the **TLS Cipher Suites for Router** and **TLS Cipher Suites for HAProxy** fields.
See also:	◦ Terminating SSL/TLS at the Load Balancer and Gorouter ◦ Providing a Certificate for Your SSL/TLS Termination Point ◦ Using the Ops Manager API

- **Disable HAProxy forwarding of requests to Router over TLS**

If you want to:	Use non-encrypted communication between HAProxy and Gorouter, or you are not using HAProxy

Then configure the following:	1. Select **Disable**. 2. If you are not using HAProxy, set the number of HAProxy job instances to `0` on the **Resource Config** page. See Disable Unused Resources.
See also:	◦ Terminating SSL/TLS at the Gorouter Only ◦ Terminating SSL/TLS at the Load Balancer Only

12. If you are not using SSL encryption or if you are using self-signed certificates, select **Disable SSL certificate verification for this environment**. Selecting this checkbox also disables SSL verification for route services.

    Note: For production deployments, Pivotal does not recommend disabling SSL certificate verification.

13. (Optional) If you want HAProxy or the Gorouter to reject any HTTP (non-encrypted) traffic, select the **Disable HTTP on HAProxy and Gorouter** checkbox. When selected, HAProxy and Gorouter will not listen on port 80.

14. (Optional) Select the **Disable insecure cookies on the Router** checkbox to set the secure flag for cookies generated by the router.

15. (Optional) To disable the addition of Zipkin tracing headers on the Gorouter, deselect the **Enable Zipkin tracing headers on the router** checkbox. Zipkin tracing headers are enabled by default. For more information about using Zipkin trace logging headers, see Zipkin Tracing in HTTP Headers.

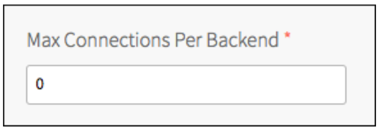
16. (Optional) To stop the Router from writing access logs to local disk, deselect the **Enable Router to write access logs locally** checkbox. You should consider disabling this checkbox for high traffic deployments since logs may not be rotated fast enough and can fill up the disk.

17. In the **Choose whether or not to enable route services** section, choose either **Enable route services** or **Disable route services**. Route services are a class of marketplace services that perform filtering or content transformation on application requests and responses. See the Route Services topic for details.

18. (Optional) If you want to limit the number of app connections to the backend, enter a value in the **Max Connections Per Backend** field. You can use this field to prevent a poorly behaving app from all the connections and impacting other apps.

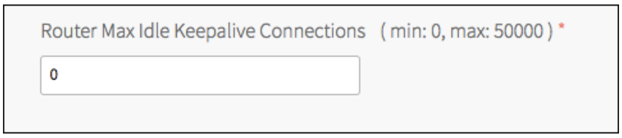
    To choose a value for this field, review the peak concurrent connections received by instances of the most popular apps in your deployment. You can determine the number of concurrent connections for an app from the `httpStartStop` event metrics emitted for each app request.

    If your deployment uses PCF Metrics, you can also obtain this peak concurrent connection information from Network Metrics. The default value is

    `500`.

19. Enter a value for **Router Max Idle Keepalive Connections**. See Considerations for Configuring max_idle_connections.

20. (Optional) To accommodate larger uploads over connections with high latency, increase the number of seconds in the **Router Timeout to Backends** field.

21. (Optional) Use the **Frontend Idle Timeout for Gorouter and HAProxy** field to help prevent connections from your load balancer to Gorouter or HAProxy from being closed prematurely. The value you enter sets the duration, in seconds, that Gorouter or HAProxy maintains an idle open connection from a load balancer that supports keep-alive.

    In general, set the value higher than your load balancer's backend idle timeout to avoid the race condition where the load balancer sends a request before it discovers that Gorouter or HAProxy has closed the connection.

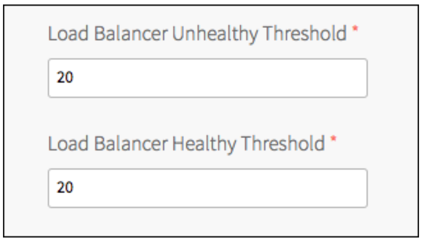
    See the following table for specific guidance and exceptions to this rule:

IaaS	Guidance
AWS	AWS ELB has a default timeout of 60 seconds, so Pivotal recommends a value greater than `60`.

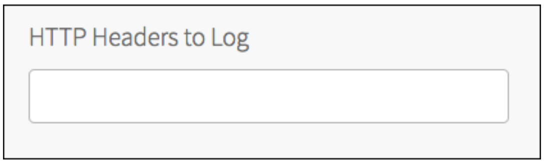
Azure	By default, Azure load balancer times out at 240 seconds without sending a TCP RST to clients, so as an exception, Pivotal recommends a value lower than `240` to force the load balancer to send the TCP RST.
GCP	GCP has a default timeout of 600 seconds, so Pivotal recommends a value greater than `600` .
Other	Set the timeout value to be greater than that of the load balancer's backend idle timeout.

22. (Optional) Increase the value of **Load Balancer Unhealthy Threshold** to specify the amount of time, in seconds, that the router continues to accept connections before shutting down. During this period, healthchecks may report the router as unhealthy, which causes load balancers to failover to other routers. Set this value to an amount greater than or equal to the maximum time it takes your load balancer to consider a router instance unhealthy, given contiguous failed healthchecks.

23. (Optional) Modify the value of **Load Balancer Healthy Threshold**. This field specifies the amount of time, in seconds, to wait until declaring the Router instance started. This allows an external load balancer time to register the Router instance as healthy.

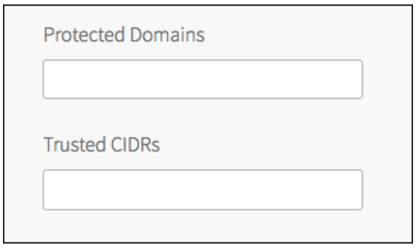


24. (Optional) If app developers in your organization want certain HTTP headers to appear in their app logs with information from the Gorouter, specify them in the **HTTP Headers to Log** field. For example, to support app developers that deploy Spring apps to PCF, you can enter Spring-specific HTTP headers.

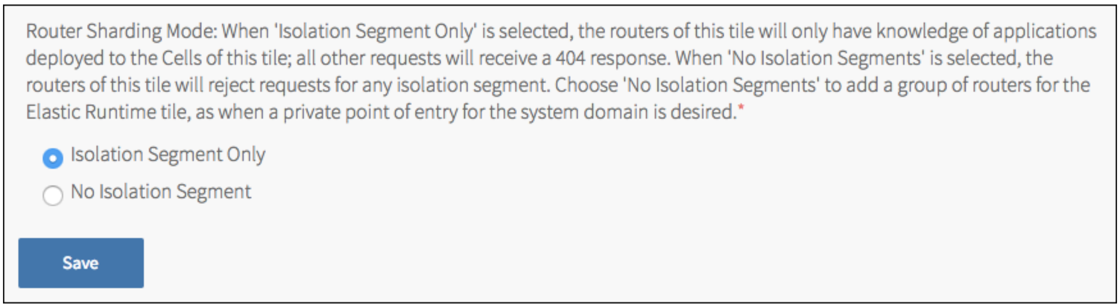


25. If you expect requests larger than the default maximum of 16 Kbytes, enter a new value (in bytes) for **HAProxy Request Max Buffer Size**. You may need to do this, for example, to support apps that embed a large cookie or query string values in headers.

26. If your PCF deployment uses HAProxy and you want it to receive traffic only from specific sources, use the following fields:
    - **Protected Domains:** Enter a comma-separated list of domains from which PCF can receive traffic.
    - **Trusted CIDRs:** Optionally, enter a space-separated list of CIDRs to limit which IP addresses from the **Protected Domains** can send traffic to PCF.



27. Select a sharding mode in the **Router Sharding Mode** field. The options are explained below. For more information, see Sharding Routers for Isolation Segments.



Option Name	Description
Isolation	

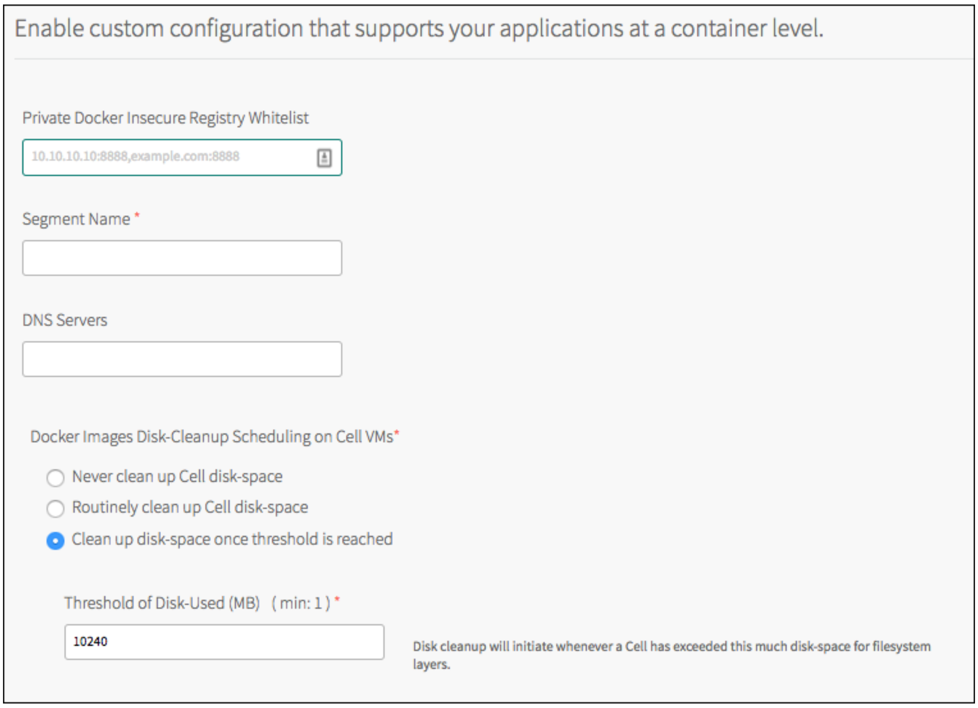
Segment Only	The routers for the tile acknowledge requests only from apps deployed within the cells of the tile. All other requests fail.
No Isolation Segment	The routers for the tile reject requests for any isolation segment. Choose this option to add a group of routers for the PAS tile, such as when you want a private point of entry for the system domain.

28. Click **Save**.

Application Containers

Perform the following steps to configure the PCF Isolation Segment tile:

1. Click **Application Containers**.



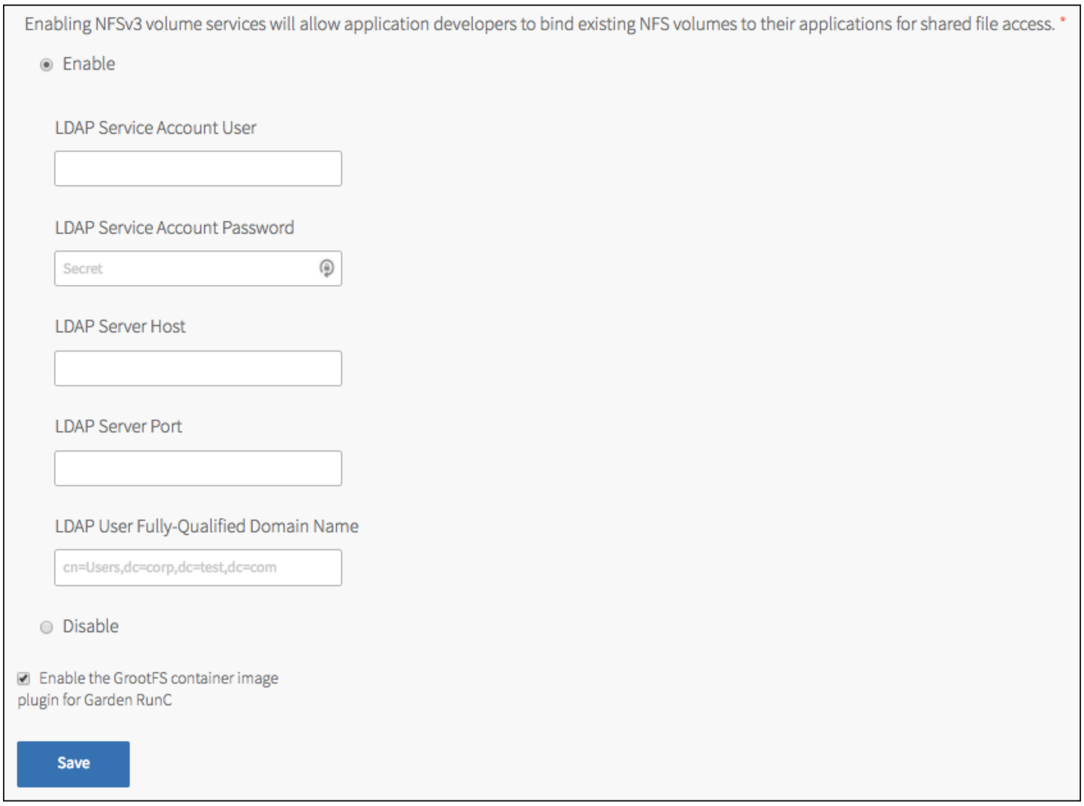
2. (Optional): Under **Private Docker Insecure Registry Whitelist**, enter one or more private Docker image registries that are secured with self-signed certificates. Use a comma-delimited list in the format `IP:Port` or `Hostname:Port`.

3. Under **Segment Name**, enter the name of your isolation segment. This name must be unique across your PCF deployment. You use this name when performing the steps in the Create an Isolation Segment section of the *Managing Isolation Segments* topic to create the isolation segment in the Cloud Controller Database (CCDB).

4. Select your preference for **Docker Images Disk-Cleanup Scheduling on Cell VMs**. If you choose **Clean up disk-space once threshold is reached**, enter a **Threshold of Disk-Used** in megabytes. For more information about the configuration options and how to configure a threshold, see Configuring Docker Images Disk-Cleanup Scheduling.

5. Under **Enabling NFSv3 volume services**, select **Enable** or **Disable**. NFS volume services allow application developers to bind existing NFS volumes to their applications for shared file access. For more information, see the Enabling NFS Volume Services topic.

> **Note:** In a clean install, NFSv3 volume services are enabled by default. In an upgrade, NFSv3 volume services match the setting of the previous deployment.



6. (Optional) To configure LDAP for NFSv3 volume services, perform the following steps:
    - For **LDAP Service Account User**, enter the username of the service account in LDAP that will manage volume services.
    - For **LDAP Service Account Password**, enter the password for the service account.
    - For **LDAP Server Host**, enter the hostname or IP address of the LDAP server.
    - For **LDAP Server Port**, enter the LDAP server port number. If you do not specify a port number, Ops Manager uses 389.
    - For **LDAP Server Protocol**, enter the server protocol. If you do not specify a protocol, Ops Manager uses TCP.
    - For **LDAP User Fully-Qualified Domain Name**, enter the fully qualified path to the LDAP service account. For example, if you have a service account called `volume-services` that belongs to organizational units (OU) called `service-accounts` and `my-company`, and your domain is called `domain`, the fully qualified path looks like the following:

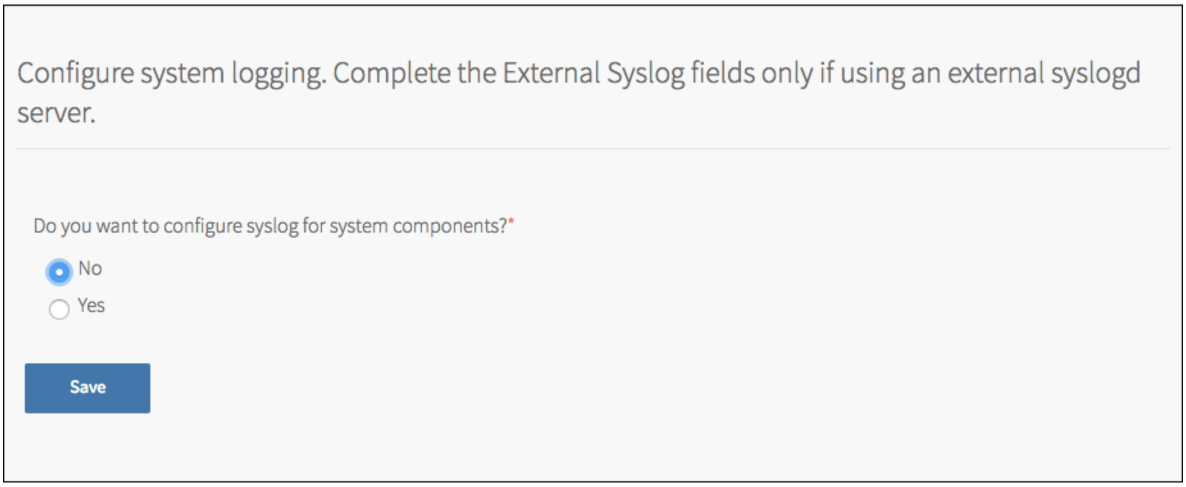
      ```
      CN=volume-services,OU=service-accounts,OU=my-company,DC=domain,DC=com
      ```

7. By default, PAS manages container images using the GrootFS plugin for Garden-runC. If you experience issues with GrootFS, you can disable the plugin and use the image plugin built into Garden-runC.

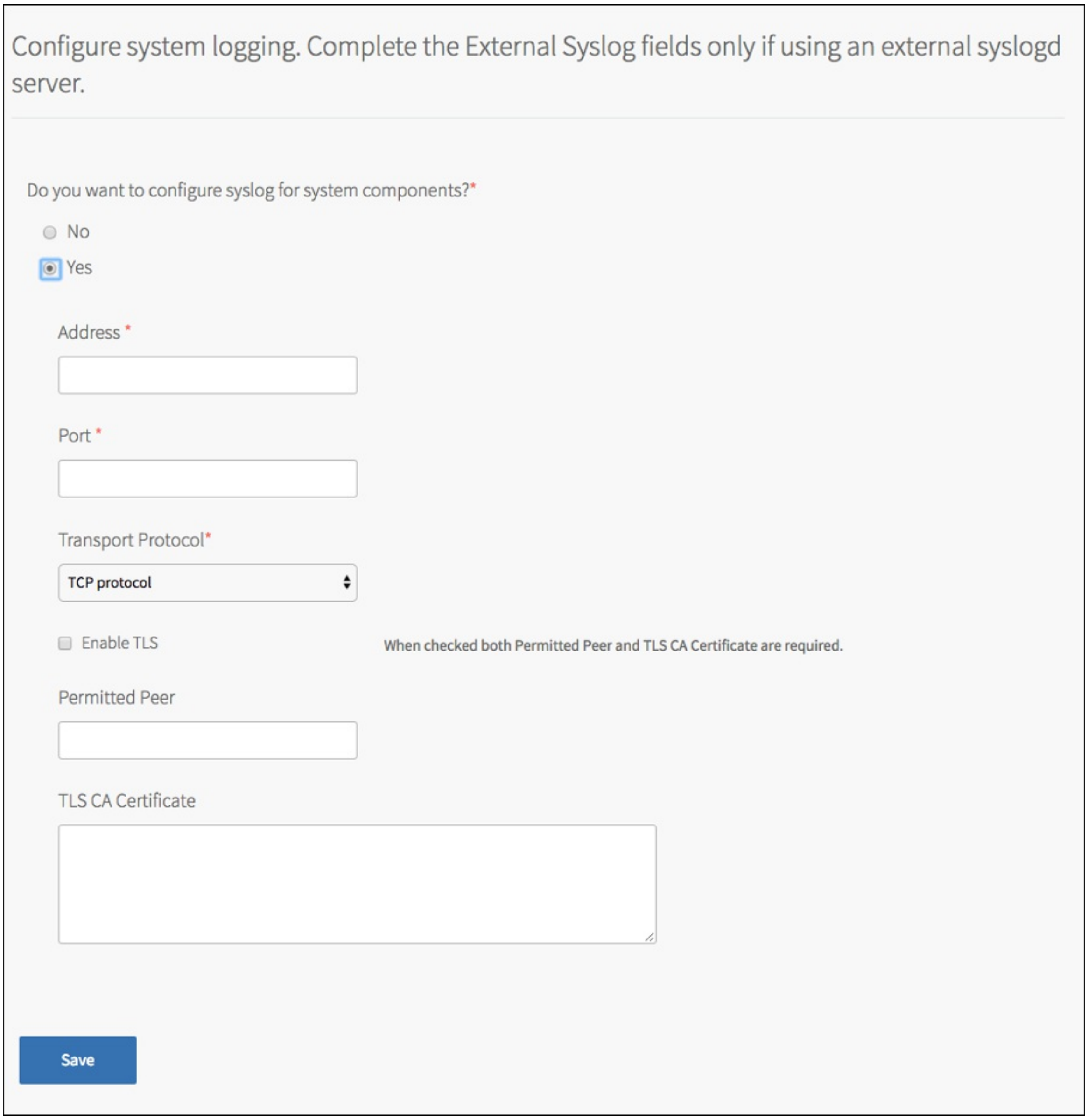
8. Click **Save**.

System Logging

1. In the **System Logging** menu, select an option from the **Do you want to configure syslog for system components?** menu. **No** is selected by default.



2. If you want to use syslog, select **Yes.**



3. Enter the address of your external syslog aggregation service in the **Address** field. The address can be a hostname or IP address.

4. Enter a port number in the **Port** field.

5. Select a protocol from the **Transport Protocol** menu. This is the protocol the sytem uses to transmit logs to syslog.

6. (Optional) Select the **Enable TLS** option if you want to transmit logs over TLS.

7. Enter a **Permitted Peer.**

8. Paste the certificate for your TLS certificate authority (CA) in the **TLS CA Certificate** field.

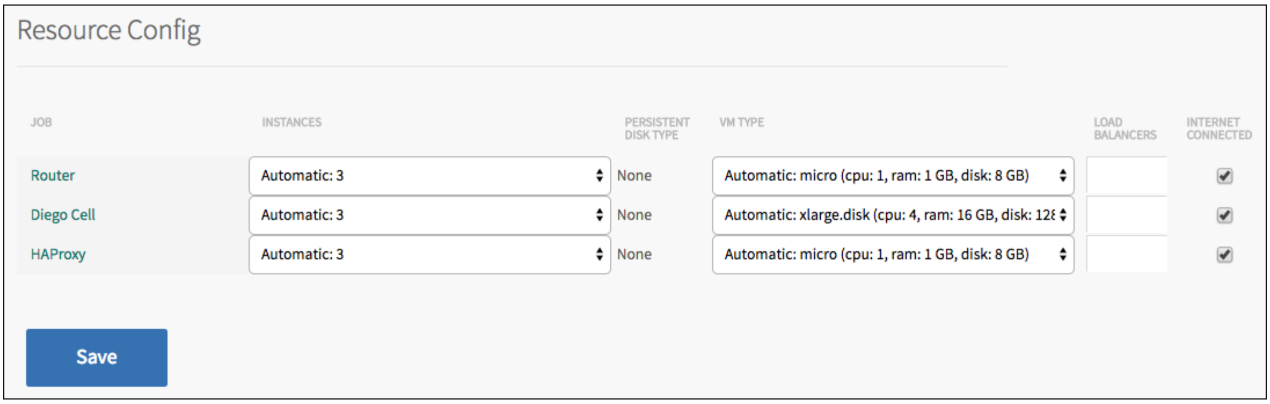
9. Click **Save.**

Advanced Features

Edit the configurations in **Advanced Features** as desired.

Configure Router Resources

1. Select **Resource Config**.



2. If you are using a dedicated router for your isolation segment, follow the instructions below.

   Note: The configuration settings available in **Resource Config** vary depending on your IaaS.

   - **If you have the Load Balancers column in your Resource Config**:
     - Enter the wildcard DNS entry attached to your load balancer into the **Router** row under **Load Balancers**.
   - **If you do not have the Load Balancers column in your Resource Config**:
     - Navigate to the **Networking** section of the **PCF Isolation Segment** tile.
     - Specify **Router IPs**.
     - Attach the IP addresses to your load balancer manually.

3. If you are not using HAProxy for this isolation segment, set the number of **Instances** to `0`.

Stemcell

Optionally, edit the **Stemcell** configuration.

After Tile Configuration

After you configure the PCF Isolation Segment tile, perform the following steps:

1. Create the isolation segment in the Cloud Controller Database (CCDB) by following the procedure in the Create an Isolation Segment section of the *Managing Isolation Segments* topic.

2. Return to the Ops Manager Installation Dashboard and click **Apply Changes** to deploy the tile.

After the tile finishes deploying, see the Managing Isolation Segments topic for more information about managing an isolation segment.

Step 4: (Optional) Copy the Tile

If you want to create multiple isolation segments, perform the following steps to copy the PCF Isolation Segment tile with the Replicator tool:

1. Download the Replicator tool from the **Pivotal Cloud Foundry Isolation Segment** section of Pivotal Network.

2. Navigate to the directory where you downloaded the Replicator tool.

3. Replicate the tile:

```
./replicator \
  -name "YOUR-NAME" \
  -path /PATH/TO/ORIGINAL.pivotal \
  -output /PATH/TO/COPY.pivotal
```

   Replace the values above with the following:
   - `YOUR-NAME`: Provide a unique name for the new PCF Isolation Segment tile. The name must be ten characters or less and only contain alphanumeric characters, dashes, underscores, and spaces.
   - `/PATH/TO/ORIGINAL`: Provide the absolute path to the original PCF Isolation Segment tile you downloaded from Pivotal Network.
   - `/PATH/TO/COPY`: Provide the absolute path for the copy that the Replicator tool produces.

4. Follow the procedures in this topic using the new `.pivotal` file, starting with Step 1.

Getting Started with Small Footprint PAS

This topic describes the Small Footprint PAS tile for Pivotal Cloud Foundry (PCF). The topic also describes how Small Footprint PAS deployment differs from a standard deployment, as well as use cases and limitations.

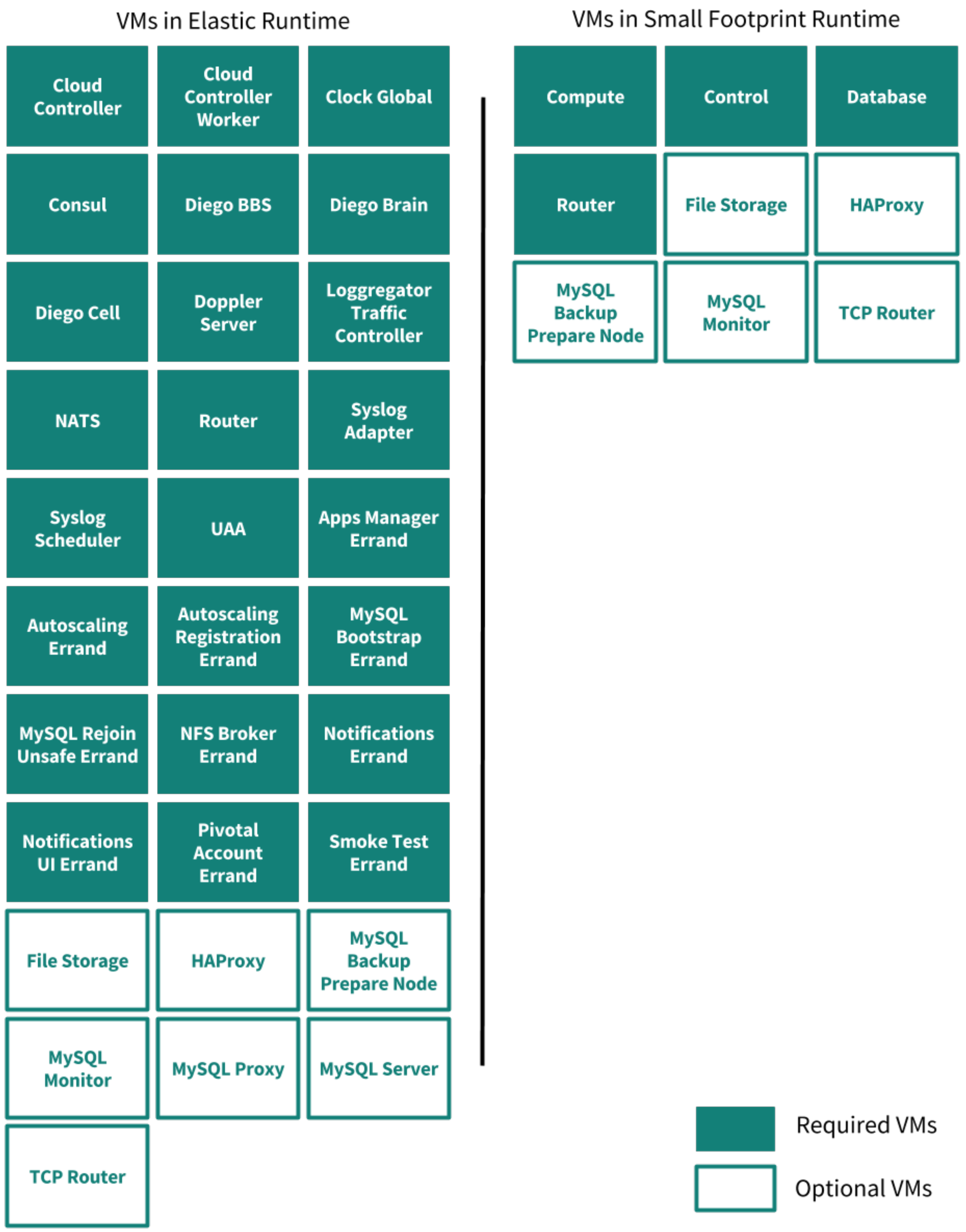
Use Small Footprint PAS for testing or small on-site data centers where performance is less important than cost.

WARNING: Pivotal does not recommend the Small Footprint PAS tile for most production deployments. The tile has significant performance and capacity restraints. See Limitations.

Differentiate Small Footprint PAS and PAS

The Small Footprint PAS is a repackaging of the Pivotal Application Service (PAS) components into a smaller and more efficient deployment. A standard PAS deployment must have at least 24 virtual machines (VMs), but the Small Footprint PAS requires only four.

The following image displays a comparison of the number of VMs deployed by PAS and Small Footprint PAS.



Use Cases

The Small Footprint PAS tile satisfies the following use cases:

- **Small on-premise installations**:
  - Run PCF in a small, on-site data center such as at a warehouse, retail store, hospital, or factory.
- **Proof-of-concept installations**:
  - Deploy PCF quickly and with a small footprint for evaluation or testing purposes.

- **Service tile R&D**:
  - Test a service tile against Small Footprint PAS instead of a standard PAS deployment to increase efficiency and reduce cost.

Limitations

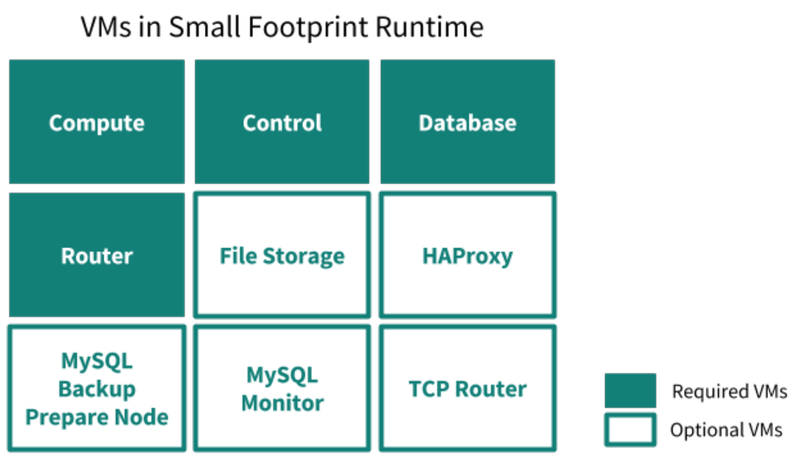
The Small Footprint PAS has the following limitations:

- **Number of app instances**:
  - The tile is not designed to support large numbers of app instances. You cannot scale the number of Compute VMs beyond `10` instances in the **Resource Config** pane. The Small Footprint PAS designed to support 2500 or fewer apps.
- **Increasing platform capacity**:
  - You cannot upgrade the Small Footprint PAS to the standard PAS tile. If you expect platform usage to increase beyond the capacity of the Small Footprint PAS, Pivotal recommends using the standard PAS tile.
- **Management plane availability during tile upgrades**
  - You may not be able to perform management plane operations like deploying new apps and accessing APIs for brief periods during tile upgrades. The management plane is colocated on the Control VM.
- **App availability during tile upgrades**:
  - If you require availability during your upgrades, you must scale your **Compute VMs** to a highly available configuration. Ensure sufficient capacity exists to move app instances between Compute VM instances during the upgrade.

Architecture

You can deploy the Small Footprint PAS tile with a minimum of four VMs, as shown in the image below.

> **Note**: The following image assumes that you are using an external blobstore.



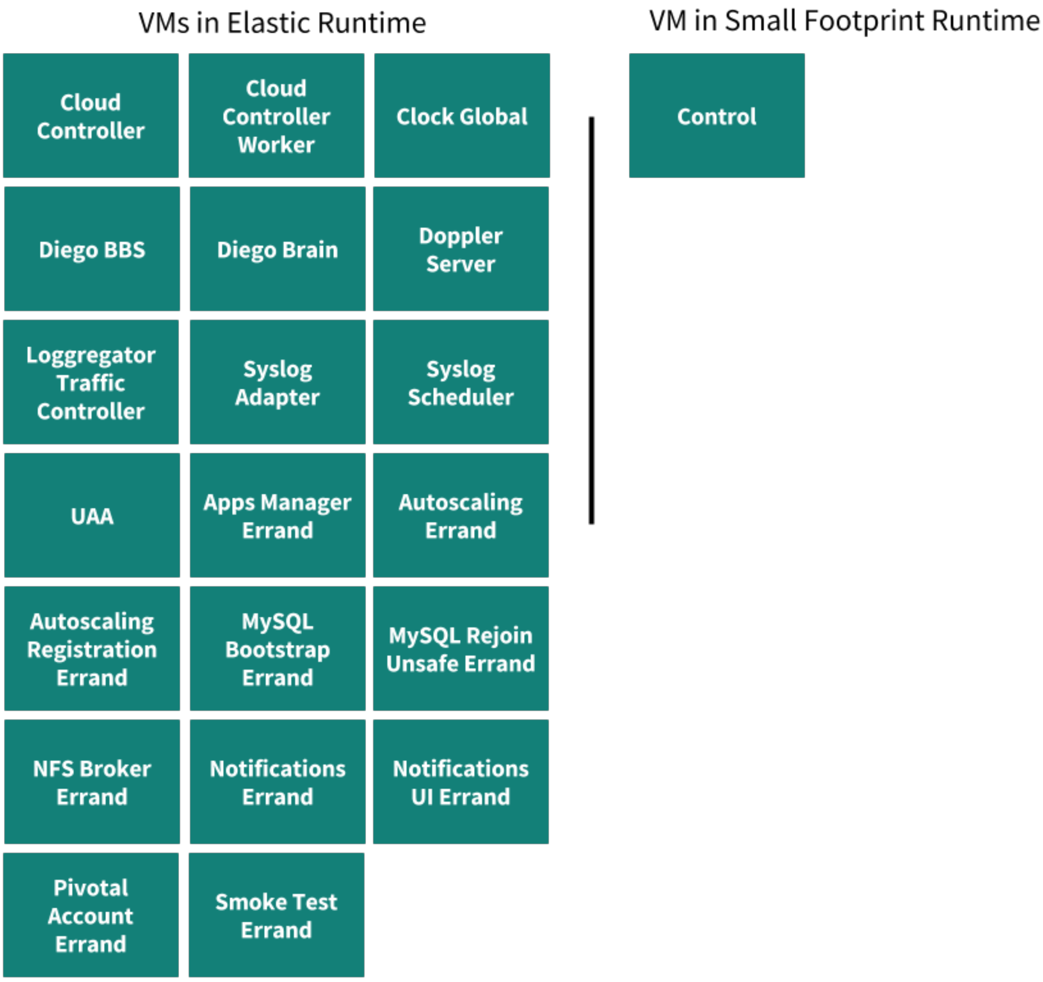
To reduce the number of VMs required for Small Footprint PAS, the *Control* and *Database* VMs include colocated jobs that run on a single VM in PAS. See the next sections for details.

For more information about the components mentioned on this page, see the *Architecture* section of the PAS Concepts guide.

Control VM

The Control VM includes the PAS jobs that handle management plane operations, app lifecycles, logging, and user authorization and authentication. Additionally, all errands run on the Control VM, eliminating the need for a VM for each errand and significantly reducing the time it takes to run errands.

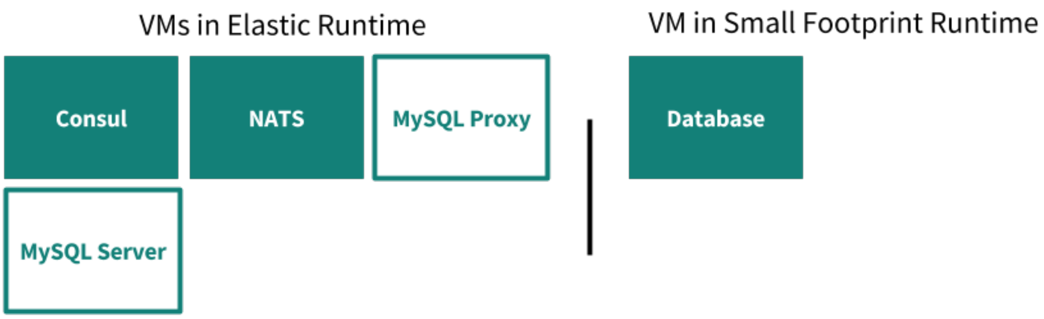
The following image shows all the jobs from PAS that are colocated on the Control VM in Small Footprint PAS.



Database VM

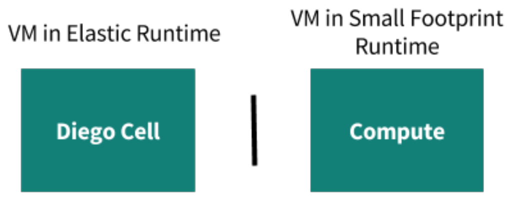
The database VM includes the PAS jobs that handle internal storage and messaging.

The following image shows all the jobs from PAS that are colocated on the Database VM in Small Footprint PAS.



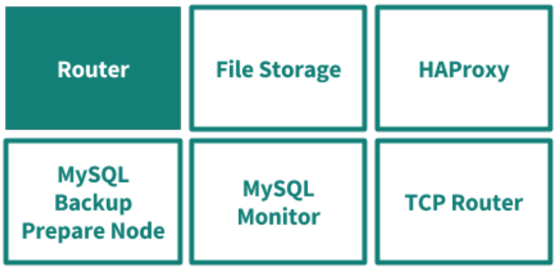
Compute VM

The *Compute* VM is the same as the Diego Cell VM in PAS.



Other VMs (Unchanged)

The following image shows the VMs performs the same functions in both versions of the PAS tile.



Installing Small Footprint PAS

To install the Small Footprint PAS tile, follow the instructions for Installing Pivotal Cloud Foundry on your IaaS.

Follow the same installation and configuration steps as for PAS, with the following differences:

- **Selecting a product in Pivotal Network**:
  - When you navigate to the PAS tile on Pivotal Network, select the **Small Footprint** release.
- **Securing communication between Diego and the Cloud Controller**:
  - In the Small Footprint PAS tile, Diego and Cloud Controller communicate securely by default.
- **Configuring resources**:
  - The **Resource Config** pane in the Small Footprint PAS tile reflects the differences in VMs discussed in the *Architecture* section of this topic.
  - Small Footprint PAS does not default to a highly available configuration like PAS does. It defaults to a minimum configuration. To make Small Footprint PAS highly available, scale the VMs to the following instance counts:
    - **Compute**: `3`
    - **Control**: `2`
    - **Database**: `3`
    - **Router**: `3`
- **Configuring load balancers**:
  - If you are using an SSH load balancer, you must enter its name in the **Control VM** row of the **Resource Config** pane. There is no **Diego Brain** row in Small Footprint Runtime because the Diego Brain is co-located on the Control VM. You can still enter the appropriate load balancers in the **Router** and **TCP Router** rows as normal.

Troubleshooting Colocated Jobs using Logs

If you need to troubleshoot a job that runs on the Control or Database VMs, follow these steps:

1. Follow the procedures in *Advanced Troubleshooting with the BOSH CLI* to the log in to the BOSH Director for your deployment:
   a. Gather Credential and IP Address Information

b. SSH into Ops Manager
c. Log in to the BOSH Director

2. Use BOSH to list the VMs in your Small Footprint PAS deployment:

```
bosh -e MY-ENV -d MY-DEPLOYMENT vms
```

Note: If you do not know the name of your deployment, you can run `bosh -e MY-ENV deployments` to list the deployments for your BOSH Director.

See the following example output:

```
$ bosh -e example-env -d example-deployment vms
Using environment 'example-env' as client 'ops_manager'


Task 182. Done


Deployment 'example-deployment'


Instance                                          Process State  AZ           IPs        VM CID                                   VM Type
backup-prepare/8fd07242-cf7c-4a4d-ba69-85fe078114f9    running       us-central1-a  10.0.4.10  vm-6ec72a47-55b0-4767-78af-759f1f295183  micro
compute/01c6947d-477e-4605-9e6b-5d130a58c70c         running       us-central1-b  10.0.4.8   vm-ce14173c-d93e-414c-6830-afbe0c713fc5  xlarge.disk
compute/28045395-5048-4c8d-8363-e22fc7b66847         running       us-central1-c  10.0.4.9   vm-e3d8f696-5802-4552-4006-a1260563ed49  xlarge.disk
compute/2e3ed7dc-baa4-42ef-814d-980c6ab1c36b         running       us-central1-a  10.0.4.7   vm-6dc34e53-71f3-4741-674a-c42c4df9e559  xlarge.disk
control/12b1b027-7ffd-43ca-9dc9-7f4ff204d86a         running       us-central1-a  10.0.4.6   vm-9760b74e-e13e-4483-79b6-78ab3818b628  xlarge
ha_proxy/b0587c68-45a8-40e2-94d3-5d2ffcdaf858        running       us-central1-a  10.0.4.11  vm-27d62bfc-af6d-4c8b-6e2a-cbba09eddd1e  micro
mysql_monitor/5185d04e-e038-4664-a26a-d16d0d295a7f    running       us-central1-a  10.0.4.15  vm-6d215888-913b-44a3-4db3-52329c5ada53  micro
router/2043b22d-0c3b-4a02-873f-80a724c3ed08          running       us-central1-a  10.0.4.12  vm-2b7cf5f4-5926-4f70-6e47-d994a6eff93b  micro
router/72b54793-e0d0-4301-8932-76da5375e654          running       us-central1-c  10.0.4.14  vm-e77bcdf1-0c26-46cd-7783-6d766f4c5098  micro
router/e3d2ab7b-6191-46bb-ab62-c1db7268a942          running       us-central1-b  10.0.4.13  vm-3e84523b-1988-475e-49e8-de80fd76c656  micro
database/681bcad5-fa8b-4cf1-912f-45140d96123f        running       us-central1-a  10.0.4.5   vm-e3cded4f-cf47-499f-4c96-992b3c6ebf9c  large.disk
tcp_router/61a06e83-a62b-4afb-b452-441dc2dc1e4c      running       us-central1-a  10.0.4.17  vm-cc1f0a62-409f-47f9-58b0-0b8f46cf9ac0  micro


13 vms


Succeeded
```

3. Use BOSH to SSH into one of the Small Footprint PAS VMs.

```
bosh -e MY-ENV -d MY-DEPLOYMENT ssh VM-NAME/GUID
```

For example, to SSH into the Control VM, run the following:

```
$ bosh -e example-env -d example-deployment ssh control/12b1b027-7ffd-43ca-9dc9-7f4ff204d86a
```

4. Run `sudo su` to act as super user.

5. Use `monit` to list the processes running on the VM.

```
monit summary
```

See the following example output that lists the processes running on the Control VM. The processes listed reflect the colocation of jobs as outlined in the *Architecture* section of this topic.

```
control/12b1b027-7ffd-43ca-9dc9-7f4ff204d86a:/var/vcap/bosh_ssh/bosh_f9d2446b18b445e# monit summary
The Monit daemon 5.2.5 uptime: 5d 21h 10m


Process 'consul_agent'            running
Process 'bbs'                     running
Process 'metron_agent'            running
Process 'locket'                  running
Process 'route_registrar'         running
Process 'policy-server'           running
Process 'silk-controller'         running
Process 'uaa'                     running
Process 'statsd_injector'         running
Process 'cloud_controller_ng'     running
Process 'cloud_controller_worker_local_1' running
Process 'cloud_controller_worker_local_2' running
Process 'nginx_cc'                running
Process 'routing-api'             running
Process 'cloud_controller_clock'  running
Process 'cloud_controller_worker_1' running
Process 'auctioneer'              running
Process 'cc_uploader'             running
Process 'file_server'             running
Process 'nsync_listener'          running
Process 'ssh_proxy'               running
Process 'tps_watcher'             running
Process 'stager'                  running
Process 'loggregator_trafficcontroller' running
Process 'reverse_log_proxy'       running
Process 'adapter'                 running
Process 'doppler'                 running
Process 'syslog_drain_binder'     running
System 'system_localhost'         running
```

6. To access logs, navigate to `/vars/vcap/sys/log` :

```
cd /var/vcap/sys/log
```

7. Run `ls` to list the log directories for each process. See the following example output from the Control VM:

```
control/12b1b027-7ffd-43ca-9dc9-7f4ff204d86a:/var/vcap/sys/log# ls
adapter     cloud_controller_clock  file_server                    nginx_cc              route_registrar  statsd_injector     uaa_ctl.err.log
auctioneer  cloud_controller_ng     locket                         nginx_newrelic_plugin routing-api      syslog_drain_binder uaa_ctl.log
bbs         cloud_controller_worker loggregator_trafficcontroller  nsync                 silk-controller  syslog_forwarder
cc_uploader consul_agent            metron_agent                   policy-server         ssh_proxy        tps
cfdot       doppler                 monit                          reverse_log_proxy     stager           uaa
```

8. Navigate to the directory of the process that you want to view logs for. For example, for the Cloud Controller process, run `cd cloud_controller_ng/` . From the directory of the process, you can list and view its logs. See the following example output:

```
control/12b1b027-7ffd-43ca-9dc9-7f4ff204d86a:/var/vcap/sys/log/cloud_controller_ng# ls
cloud_controller_ng_ctl.err.log  cloud_controller_ng.log.2.gz  cloud_controller_ng.log.6.gz          drain                pre-start.stdout.log
cloud_controller_ng_ctl.log      cloud_controller_ng.log.3.gz  cloud_controller_ng.log.7.gz          post-start.stderr.log
cloud_controller_ng.log          cloud_controller_ng.log.4.gz  cloud_controller_worker_ctl.err.log   post-start.stdout.log
cloud_controller_ng.log.1.gz     cloud_controller_ng.log.5.gz  cloud_controller_worker_ctl.log       pre-start.stderr.log
```

Release Notes

The Small Footprint PAS tile releases alongside the PAS tile. See the PAS Release Notes.

Upgrading Pivotal Cloud Foundry

Page last updated:

Note: Elastic Runtime has been renamed Pivotal Application Service.

This topic describes upgrading Pivotal Cloud Foundry (PCF) to v2.0. The upgrade procedure below describes upgrading Pivotal Cloud Foundry Operations Manager (Ops Manager), Pivotal Application Service (PAS), and product tiles.

The apps in your deployment continue to run during the upgrade. However, you cannot write to your deployment or make changes to apps during the upgrade.

Important: Read the Release Notes and Breaking Changes for this release, including the Known Issues sections, before starting the upgrade process.

For more details about the impact of upgrading on individual PCF components, see the Understanding the Effects of Single Components on a Pivotal Cloud Foundry Upgrade topic.

Before You Upgrade

This section contains important preparation steps, **Preps**, that you must follow before beginning an upgrade to Ops Manager v2.0. Failure to follow these instructions may jeopardize your existing deployment data and cause your upgrade to fail.

Pivotal recommends backing up your PCF deployment before upgrading, to restore in the case of failure. To do this, follow the instructions in the Backing Up Pivotal Cloud Foundry with BBR topic.

Breaking Change: If you have the Single Sign-On (SSO) service tile installed, you must first upgrade the SSO tile to v1.5.3 and configure the Apps Manager errand before you can upgrade the PAS tile to 2.0. To properly upgrade your SSO tile, see Upgrade SSO Service Tile resource instance between PCF 1.12 and PCF 2.0 in the Pivotal Knowledge Base.

Prep 1: Review File Storage IOPS and Other Upgrade Limiting Factors

During the PCF upgrade process, a large quantity of data is moved around on disk.

To ensure a successful upgrade of PCF, verify that your underlying PAS file storage is performant enough to handle the upgrade. For more information about the configurations to evaluate, see Upgrade Considerations for Selecting Pivotal Cloud Foundry Storage.

In addition to file storage IOPS, consider additional existing deployment factors that can impact overall upgrade duration and performance:

Factor	Impact
Network latency	Network latency can contribute to how long it takes to move app instance data to new containers.
Number of ASGs	A large number of Application Security Groups in your deployment can contribute to an increase in app instance container startup time.
Number of app instances and application growth	A large increase in the number of app instances and average droplet size since the initial deployment can increase the upgrade impact on your system.

To review example upgrade-related performance measurements of an existing production Cloud Foundry deployment, see the Pivotal Web Services Performance During Upgrade topic.

Prep 2: Verify App Usage Service Remedial Steps

If you are upgrading from a PCF deployment that at one point included Elastic Runtime v1.7.16 or earlier, you must perform the remedial steps outlined in App Usage Data and Events Data Become Corrupted After Upgrade or Install before proceeding with this upgrade.

Warning: If you fail to perform the remedial steps for this issue, this upgrade process may corrupt your existing usage data.

Prep 3: Check Certificate Authority Expiration Dates

Depending on the requirements of your deployment, you may need to rotate your Certificate Authority (CA) certificates. The non-configurable certificates in your deployment expire every two years. You must regenerate and rotate them so that critical components do not face a complete outage.

> **Note**: Pivotal Cloud Foundry (PCF) uses SHA-2 certificates and hashes by default. You can convert existing SHA-1 hashes into SHA-2 hashes by rotating your Ops Manager certificates using the procedure described in Regenerating and Rotating Non-Configurable TLS/SSL Certificates.

On the command line, run `GET /api/v0/deployed/certificates?expires_within=TIME` to retrieve information about all the RSA and CA certificates for your deployment, including whether they are set to expire. Replace `TIME` with an integer and a letter code. Valid letter codes are `d` for days, `w` for weeks, `m` for months, and `y` for years.

For example, to search for certificates expiring within one month, replace `TIME` with `1m` as follows:

```
$ GET /api/v0/deployed/certificates?expires_within=1m
```

For information about how to regenerate and rotate CA certificates, see Managing Non-Configurable TLS/SSL Certificates.

Prep 4: Review Partner Service Tiles

Some partner service tiles may be incompatible with PCF v2.0. Pivotal is working with partners to ensure their tiles are updated to work with the latest versions of PCF. For information about which partner service releases are currently compatible with PCF v2.0, review the appropriate partners services release documentation at http://docs.pivotal.io, or contact the partner organization that produces the service tile.

Prep 5: Download Upgrade Versions

To minimize disruptions to your deployment during the upgrade, and to satisfy any simultaneous upgrade requirements, download the version of the product files you wish to upgrade from Pivotal Network.

At the minimum, you must download PAS v2.0.x.

Prep 6: Prepare Your Environment

1. Install the releases from your currently deployed version to the target version in sequential order. For example, if your deployment uses Ops Manager v1.8 and you are upgrading to v2.0, you must sequentially install v1.9, v1.10, v1.11, and v1.12 before proceeding with the upgrade to v2.0.

2. If you have disabled lifecycle errands for any installed product to reduce deployment time, Pivotal recommends that you re-enable these errands before upgrading. For more information, see the Adding and Deleting Products topic.

3. Confirm that you have adequate disk space for your upgrades. You need at least 20 GB of free disk space to upgrade PCF Ops Manager and Pivotal Application Service (PAS). If you plan to upgrade other products, the amount of disk space required depends on how many tiles you plan to deploy to your upgraded PCF deployment.

   To check current persistent disk usage, select the **Ops Manager Director** tile from the **Installation Dashboard**. Select **Status** and review the value of the `PERS. DISK` column. If persistent disk usage is higher than 50%, select **Settings > Resource Config**, and increase your persistent disk space to handle the size of the resources. If you do not know how much disk space to allocate, set the value to at least `100 GB`.

4. If not already disabled, disable the VM Resurrector:

   a. From your **Installation Dashboard**, select the **Ops Manager Director** tile.
   b. Click **Director Config**.
   c. Clear the **Enable VM resurrector plugin** checkbox.
   d. Click **Save**.
   e. Return to the **Installation Dashboard** and click **Apply Changes**.

5. If your original PAS deployment was PCF 1.6 or earlier, rotate non-configurable Director certificates using the Ops Manager API. Follow Step 3 of the Managing Non-Configurable TLS/SSL Certificates topic to regenerate certificates, or begin with Step 1 to also update the certificate authority (CA).

6. Check the required machine specifications for Ops Manager v2.0. These specifications are specific to your IaaS. If these specifications do not match your existing Ops Manager, modify the values of your Ops Manager VM instance. For example, if the boot disk of your existing Ops Manager is 50 GB and the new Ops Manager requires 100 GB, then increase the size of your Ops Manager boot disk to 100 GB.

7. If you are upgrading a vSphere environment, ensure that you have the following information about your existing environment before starting the upgrade:
   - Record the following IP addresses, which you can find in the vSphere web client under **Manage** > **Settings** > **vApp Options**. This is the same information you entered at the end of deploying Ops Manager on vSphere.
     - IP Address of the Ops Manager
     - Netmask
     - Default Gateway
     - DNS Servers
     - NTP Servers
   - Record the following VM hardware information so you can configure the new VM with similar settings. You can find this information in the vSphere web client under **Manage** > **Settings** > **VM Hardware**.
     - CPU
     - Memory
     - Hard Disk 1
     - Network Adapter 1. When you configure the new VM, ensure your network adapters are configured properly and are on the same network.

Prep 7: Upgrade MySQL for PCF

If your PCF deployment includes both **PAS for Windows 2012R2** and **MySQL for PCF v1.x**, download and upgrade to MySQL for PCF v1.10.6 or later. For instructions on how to upgrade MySQL for PCF, see the MySQL for PCF documentation.

Prep 8: Upgrade and Configure RabbitMQ for PCF

If your PCF deployment contains **RabbitMQ for PCF**, download and upgrade RabbitMQ for PCF to v1.11 or later. For upgrade instructions, see the RabbitMQ for PCF documentation. As part of the upgrade, ensure that the firewall rules on your Rabbit VM instances allow inbound traffic on port `8301`.

Prep 9: Upgrade and Configure Redis for PCF

If your PCF deployment contains **Redis for PCF**, download and upgrade to Redis for PCF v1.10. Ensure that you complete the following as part of the upgrade:

- When performing the upgrade, ensure that persistent disk is set to 3.5x the amount of RAM for your Dedicated-VM Redis instances. See Networks, Security, and Assigning AZs for more information.
- Ensure you have configured firewall rules on your Redis VM instances.

For upgrade instructions, see the Redis for PCF documentation.

Prep 10: Check OS Compatibility of PCF and BOSH-Managed Add-Ons

Before upgrading to PCF v2.0, operators who have deployed any PCF add-ons (such as **ClamAV for PCF**, **IPsec for PCF**, or **File Integrity Monitoring for PCF**) and who have deployed or are planning to deploy PAS for Windows 2012R2 must modify the add-on manifest to specify a compatible OS stemcell.

For example, **ClamAV for PCF** is not supported on Windows. Therefore, the manifest must use an `include` directive to specify the target OS stemcell of `ubuntu-trusty`.

To update an add-on manifest, perform the following steps:

1. Locate your existing add-on manifest file. For example, for ClamAV, locate the `clamav.yml` you uploaded to the Ops Manager VM.
2. Modify the manifest to include following `include` directive to your manifest:

```
include:
  stemcell:
    - os: ubuntu-trusty
```

3. Re-upload the manifest file to your PCF deployment. For example instructions, see Create the ClamAV Manifest.

If you are using any other BOSH-managed add-ons in your deployment, you should verify OS compatibility for those component as well. For more information about configuring BOSH add-on manifests, see the BOSH documentation.

Prep 11: Check System Health Before Upgrade

1. Run `bosh cloud-check` to confirm that the VMs are healthy. For more information, see the BOSH Cloudcheck topic.

2. Check the system health of installed products. In the **Installation Dashboard**, select the **Status** tab for each service tile. Confirm that all jobs are healthy.

3. (Optional) Check the logs for errors before proceeding with the upgrade. For more information, see the Viewing Logs in the Command Line Interface topic.

4. Confirm there are no outstanding changes in Ops Manager or any other tile. All tiles should be **green**. Click **Apply Changes** if necessary.

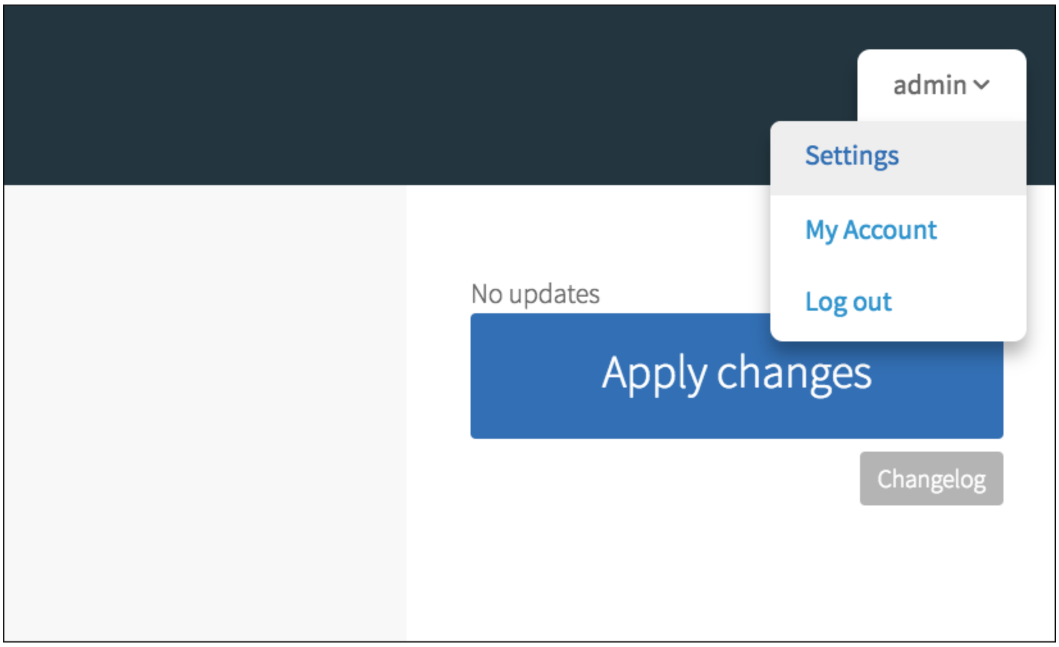
5. After applying changes, click **Recent Install Logs** to confirm that the changes completed cleanly:

```
Cleanup complete
{"type": "step_finished", "id": "clean_up_bosh.cleaning_up"}
Exited with 0.
```

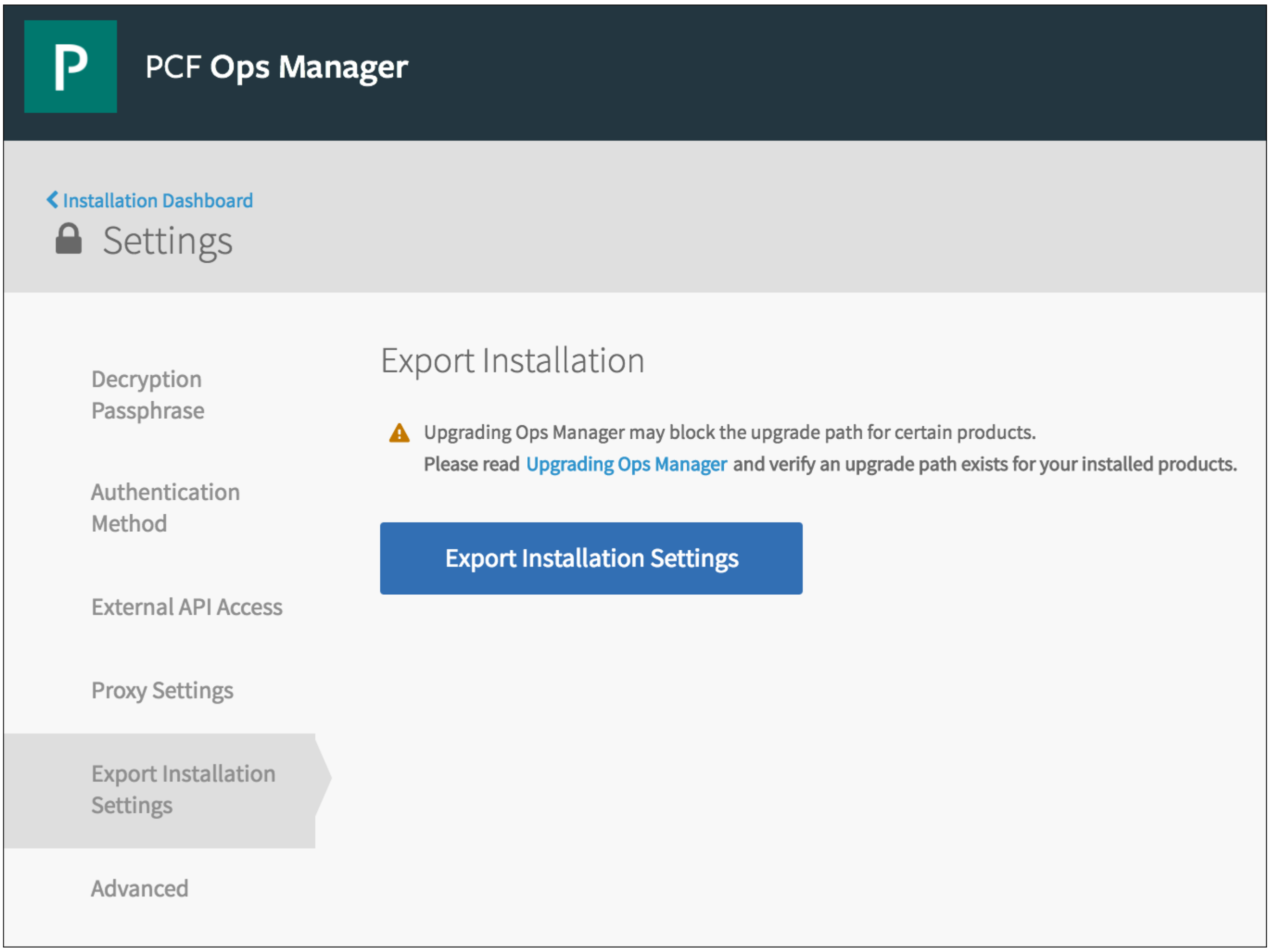
Upgrade Ops Manager and Installed Products to v2.0

Step 1: Export Your Installation

1. In your Ops Manager v1.12.x **Installation Dashboard**, click the account dropdown and select **Settings**.



2. On the **Settings** screen, select **Export Installation Settings** from the left menu, then click **Export Installation Settings**.

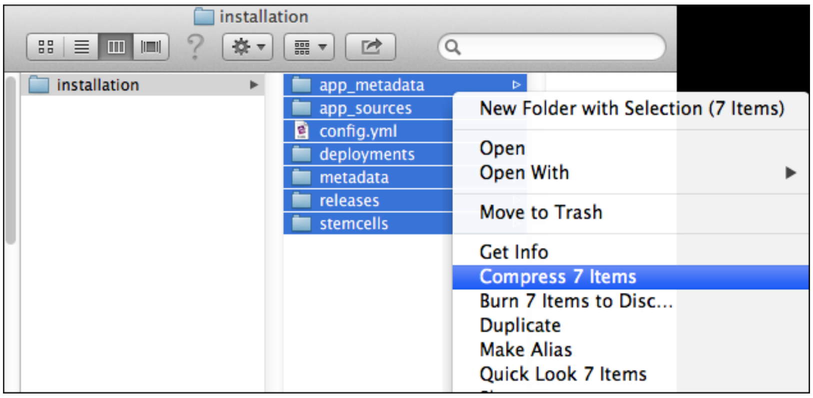


This exports the current PCF installation with all of its assets.

When you export an installation, the export contains the base VM images and necessary packages, and references to the installation IP addresses. As a result, an exported file can be very large, 5 GB or more.

- The export time depends on the size of the exported file.
- Some browsers do not provide feedback on the status of the export process and might appear to hang.

Note: Some operating systems automatically unzip the exported installation. If this occurs, create a ZIP file of the unzipped export. Do not start compressing at the "installation" folder level. Instead, start compressing at the level containing the `config.yml` file:



Step 2: Upgrade to Ops Manager v2.0

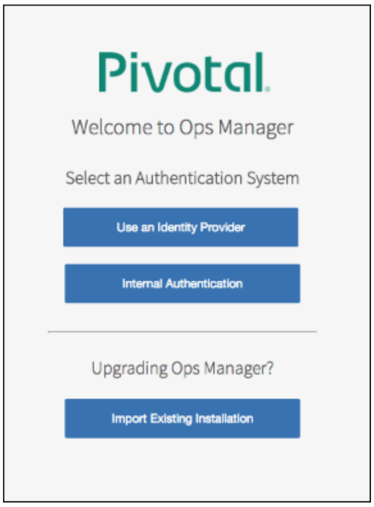
1. Download the Ops Manager VM Template v2.0.x from the Pivotal Network site.

2. Record the FQDN address of the existing Ops Manager VM.

3. To avoid conflicts, power off the existing Ops Manager VM.

4. Deploy the new Ops Manager VM by following the steps in one of these topics:
   - **AWS (Terraform)**: Deploying BOSH and Ops Manager to AWS and subsequent topics.
   - **AWS (Manual)**: Manually Configuring AWS for PCF
   - **Azure**: Upgrading Ops Manager Director on Azure
   - **GCP**: Upgrading Ops Manager Director on GCP
   - **OpenStack**: Provisioning the OpenStack Infrastructure
   - **vSphere**: Deploying Operations Manager to vSphere

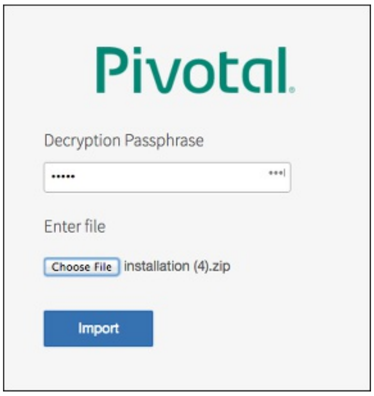
5. When redirected to the **Welcome to Ops Manager** page, select **Import Existing Installation**.



6. When prompted, enter a **Decryption Passphrase**.

> **Note**: Record and store your **Decryption Passphrase** in a safe location. If lost, the **Decryption Passphrase** cannot be recovered.

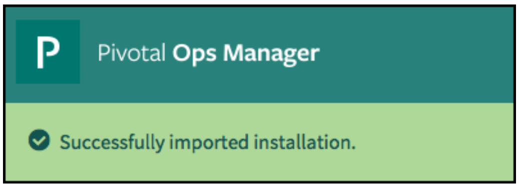
7. Click **Choose File** and browse to the installation ZIP file exported in Step 1 above.



8. Click **Import**.

> **Note:** Some browsers do not provide feedback on the status of the import process, and might appear to hang.

9. A “Successfully imported installation” message appears upon completion.



Step 3: Upgrade PAS and Product Tiles

1. After upgrading to Ops Manager v2.0, upgrade your product versions.
2. Import the product file to your Ops Manager **Installation Dashboard**.
3. Hover over the product name in **Available Products** and click **Add**.
4. Click the newly added tile to review any configurable options.
5. (Optional) If you are using other service tiles, you can upgrade them following the same procedure. See the Upgrading PAS and Other Pivotal Cloud Foundry Products topic for more information.

Step 4: Perform Your Upgrade

1. Navigate to the Ops Manager **Installation Dashboard**.
2. Click **Apply Changes**. This immediately imports and applies upgrades to all tiles in a single transaction.

   WARNING: If the installation fails or returns errors, contact Support. Do not attempt to roll back the upgrade by restarting the previous (v1.12.x) Ops Manager VM.

3. Click each service tile, select the **Status** tab, and confirm that all VMs appear and are in good health.
4. After confirming that the new installation functions correctly, remove the previous (v1.12.x) Ops Manager VM.

Step 5: Complete Your Installation

1. Navigate to the Ops Manager **Installation Dashboard**.
2. Click **Apply Changes**. This immediately imports and applies upgrades to all tiles in a single transaction.

   WARNING: If the installation fails or returns errors, contact Support. Do not attempt to roll back the upgrade by restarting the previous (v1.12.x) Ops Manager VM.

3. Click each service tile, select the **Status** tab, and confirm that all VMs appear and are in good health.

After you Upgrade

Upgrade cf CLI

To use the experimental push commands, multiple buildpack support, and container networking commands in PCF v2.0, make sure you have the latest version of cf CLI installed.

Understanding the Effects of Single Components on a Pivotal Cloud Foundry Upgrade

Page last updated:

The **Resource Config** page of Pivotal Application Service (PAS) tile in the Pivotal Cloud Foundry (PCF) Ops Manager shows the components that the Ops Manager Director installs. You can specify the number of instances for some of the components. We deliver the remaining resources as single components, meaning that they have a preconfigured and unchangeable value of one instance.

In a single-component environment, upgrading can cause the deployment to experience downtime and other limitations because there is no instance redundancy. Although this behavior might be acceptable for a test environment, you should configure the scalable components with editable instance values, such as HAProxy, Router, and Diego cells, for optimal performance in a production environment.

Note: A full Ops Manager upgrade may take close to two hours, and you will have limited ability to deploy an application during this time.

Summary of Component Limitations

The table lists components in the order that Ops Manager upgrades each component and includes the following columns:

- **Scalable?**: Indicates whether the component has an editable value or a preconfigured and unchangeable value of one instance.

  Note: For components marked with a checkmark in this column, we recommend that you change the preconfigured instance value of 1 to a value that best supports your production environment. For more information about scaling a deployment, refer to the Scaling Cloud Foundry topic.

- **Extended Downtime?**: Indicates that if there is only one instance of the component, that component is unavailable for up to five minutes during an Ops Manager upgrade.
- **Other Limitations and Information**: Provides the following information:
  - Component availability, behavior, and usage during an upgrade
  - Guidance on disabling the component before an upgrade

Note: The table does not include the Run Smoke Tests and Run CF Acceptance Tests errands and the Compilation job. Ops Manager runs the errands after it upgrades the components and creates compilation VMs as needed during the upgrade process.

Upgrade Order	Component	Scalable?	Extended Downtime?	Other Limitations and Information
1	Consul	✓	✓	Many components rely upon Consul for service discovery. If Consul is unavailable, these components may fail in unexpected ways as they are not able to locate or communicate with other parts of the platform.
2	NATS	✓	✓	
4	File Storage		✓	You cannot push, stage, or restart an app when an upgrade affects the file storage server.
5	MySQL Proxy	✓	✓	The MySQL Proxy is responsible for managing failover of the MySQL Servers. If the Proxy becomes unavailable, then access to the MySQL Server could be broken.
6	MySQL Server	✓	✓	The MySQL Server is responsible for persisting internal databases for the platform. If the MySQL Server becomes unavailable, then platform services that rely upon a database (Cloud Controller, UAA) will also become unavailable.
7	Backup Prepare Node			
10	UAA	✓		If a user has an active authorization token prior to performing an upgrade, the user can still log in using either a UI or the CLI.
11	Cloud Controller	✓	✓	Your ability to manage an app when an upgrade affects the Cloud Controller depends on the number of instances that you specify for the Cloud Controller and Diego components. If either of these components are single components, you cannot push, stage, or restart an app during the upgrade.
12	HAProxy	✓	✓	HAProxy is used to load-balance incoming requests to the Router. If HAProxy is unavailable, you may lose the ability to make requests to applications unless there is another routing path from your load balancer to the Router.

				from your load balancer to the Router.
13	Router	✓	✓	The Router is responsible for routing requests to their application containers. If the Router is not available, then applications cannot receive requests.
14	MySQL Monitor			
15	Clock Global	✓		
16	Cloud Controller Worker	✓	✓	
17	Diego BBS	✓	✓	Your ability to manage an app when an upgrade affects the Diego BBS depends on the number of instances that you specify for the Diego BBS, Cloud Controller, and other Diego components. If any of these components have only one instance, you may fail to push, stage, or restart an app during the upgrade.
18	Diego Brain	✓	✓	Your ability to manage an app when an upgrade affects the Diego Brain depends on the number of instances that you specify for the Diego Brain, Cloud Controller, and other Diego components. If any of these components have only one instance, you may fail to push, stage, or restart an app during the upgrade.
19	Diego Cell	✓	✓	Your ability to manage an app when an upgrade affects Diego Cells depends on the number of instances that you specify for the Diego Cells, Cloud Controller, and other Diego components. If any of these components have only one instance, you may fail to push, stage, or restart an app during the upgrade. If you only have one Diego Cell, upgrading it causes downtime for the apps that run on it, including the Apps Manager app and the App Usage Service.
20	Doppler Server	✓		Ops Manager operators experience 2-5 minute gaps in logging.
21	Loggregator Trafficcontroller	✓		Ops Manager operators experience 2-5 minute gaps in logging.
22	TCP Router	✓		
23	Push Apps Manager errand			This errand runs the script to deploy the Apps Manager application. The Apps Manager application runs in a single Diego Cell.
24	Run Smoke Tests			
25	Push Notifications			
26	Push Notifications UI			
27	Push Autoscaling			
28	Register Autoscaling Service Broker			
29	Destroy Autoscaling Service Broker			
30	Bootstrap			
31	Push Pivotal Account			
32	MySQL Rejoin Unsafe Errand			

Upgrade Considerations for Selecting File Storage in Pivotal Cloud Foundry

Page last updated:

This topic describes critical factors to consider when evaluating the type of file storage to use in your Pivotal Cloud Foundry (PCF) deployment. The Pivotal Application Service (PAS) blobstore relies on the file storage system to read and write resources, app packages, and droplets.

During an upgrade of PCF, file storage with insufficient IOPS numbers can negatively impact the performance and stability of your PCF deployment.

If disk processing time takes longer than the evacuation timeout for Diego cells, then Diego cells and app instances may take too long to start up, resulting in a cascading failure.

However, the minimum required IOPS depends upon a number of deployment-specific factors and configuration choices. Use this topic as a guide when deciding on the file storage configuration for your deployment.

To see an example of system performance and IOPS load during an upgrade, refer to Upgrade Load Example: Pivotal Web Services.

Selecting Internal or External File Storage

When you deploy PCF, you can select internal file storage or external file storage, either network-accessible or IaaS-provided, as an option in the PAS tile.

Selecting internal storage causes PCF to deploy a dedicated virtual machine (VM) that uses either NFS or WebDAV for file storage. Selecting external storage allows you to configure file storage provided in network-accessible location or by an IaaS, such as Amazon S3, Google Cloud Storage, or Azure Storage.

Whenever possible, Pivotal recommends using external file storage.

Calculating Potential Disk Load Requirements

As a best-effort calculation, estimate the total number of bits needed to move during a system upgrade to determine how IOPS-performant your file storage needs to be.

Number of Diego Cells

As a first calculation, determine the number of Diego cells that your deployment currently uses.

To view the number of Diego cell instances currently running in your deployment, see the **Resource Config** section of your PAS tile.

If you expect to scale up the number of instances, use the anticipated scaled number.

> **Note**: If your deployment uses more than 20 Diego cells, you should avoid using internal file storage. Instead, you should always select external or IaaS-provided file storage.

Maximum In-Flight Load and Container Starts for Diego Cells

Operators can limit the number of containers and Diego cell instances that Diego starts concurrently. If operators impose no limits, your file storage may experience exceptionally heavy load during an upgrade.

To prevent overload, Cloud Foundry provides two major throttle configurations:

- **The maximum number of starting containers that Diego can start in Cloud Foundry**: This is a deployment-wide limit. The default value and ability to override this configuration depends on the version of Cloud Foundry deployed. For information about how to configure this setting, see the Setting a Maximum Number of Started Containers topic.
- **The `max_in_flight` setting for the Diego cell job configured in the BOSH manifest**: This configuration, expressed as a percentage or an integer, sets the maximum number of job instances that can be upgraded simultaneously. For example, if your deployment is running 10 Diego cell job instances and the configured `max_in_flight` value is `20%`, then only 2 Diego cell job instances can start up at a single time.

  To retrieve or override the existing `max_in_flight` value in Ops Manager Director, use the Ops Manager API. See the Ops Manager API documentation provided with your Ops Manager installation at `https://YOUR-OPSMAN-FQDN/docs/` .

The values of the above throttle configurations depend on the version of PCF that you have deployed and whether you have overridden the default values.

Refer to the following table for existing defaults and, if necessary, determine the override values in your deployment.

PCF Version	Starting Container Count Maximum	Starting Container Count Overridable?	Maximum In Flight Diego Cell Instances	Maximum In Flight Diego Cell Instances Overridable?
PCF 1.7.43 and earlier	No limit set	No	1 instance	No
PCF 1.7.44 to 1.7.49	200	No	1 instance	No
PCF 1.7.50 +	200	No	1 instance	No
PCF 1.8.0 to 1.8.29	No limit set	No	10% of total instances	No
PCF 1.8.30 +	200	Yes	10% of total instances	No
PCF 1.9.0 to 1.9.7	No limit set	No	4% of total instances	Yes
PCF 1.9.8 +	200	Yes	4% of total instances	Yes
PCF 1.10.0 and later	200	Yes	4% of total instances	Yes
PCF 1.12.0 and later	200	Yes	4% of total instances	Yes

Calculating Upgrade Load Based on Number of App Instances and Droplet Size

Using the above numbers, you can determine a rough estimate of the expected upgrade load by multiplying the total number of expected app instances for all cells with the size of the instance droplets.

For example, if your deployment starts 10 cells that each host 20 app instances, and each app instance droplet is an average of 100 MB in size, then you potentially have 20 GB of data hitting the disk at the same time. Depending on the IOPS capacity of your disk, this 20 GB of data will take a set amount of time to reassemble on a new disk.

Calculate the amount of time needed to process your potential upgrade load, and verify that the number falls under the evacuation timeout (default is 10 minutes) for Diego cells.

If the calculated processing time is longer than the evacuation timeout, you should upgrade your file storage to use disk with higher IOPS capacity.

For more information about how Diego cells are upgraded, see the Managing Diego Cell Limits During an Upgrade topic.

Related Links

- How to use PAS blob storage data
- Upgrading Pivotal Cloud Foundry
- Managing Diego Cell Limits During an Upgrade

Upgrade Load Example: Pivotal Web Services

Page last updated:

This topic provides sample performance measurements of a Cloud Foundry installation undergoing the workload associated with an upgrade.

To obtain these measurements, Pivotal repaved its production Pivotal Web Services (PWS) deployment. The repave process simulates system load that would be incurred when performing a rolling upgrade of Diego cells.

Use the measurements and configuration values published in this document as guidance when ensuring you have adequate file storage hardware prior to a platform upgrade.

For more information about the impact of upgrade on file storage performance, see Upgrade Considerations for Selecting File Storage in Pivotal Cloud Foundry.

Platform Configuration

The following table details the starting parameters and configuration of PWS.

Configuration	Value	How to Locate
IaaS	Amazon Web Services	Refer to your Ops Manager Director configuration or BOSH deployment manifest.
File Storage	AWS EBS (External with some elastic capacity)	Refer to your Pivotal Application Service (PAS) configuration or BOSH deployment manifest.
Version of CF	v252	Refer to your Ops Manager Director and PAS configuration or BOSH deployment manifest.
Number of Diego Cells	218	To view the number of Diego cell instances currently running in your deployment, see the**Resource Config** section of your PAS tile or consult your Diego deployment manifest.
Maximum Number of Started Containers	250	See PCF or Cloud Foundry documentation for configuration information.
max_in_flight Configuration for Diego Cells	6	To retrieve the existing `max_in_flight` value for the Diego Cell job in Ops Manager Director, use the Ops Manager API. See the Ops Manager API documentation. If you are running open source CF, consult your BOSH deployment manifest.
Number of Availability Zones (AZ)	2	Consult your PAS or BOSH deployment AZ configuration.
Number of App Instances	16231	`datadog.nozzle.bbs.LRPsRunning`
Number of Application Security Groups (ASGs)	43	As admin user, run the `cf security-groups` command. For more information, see Understanding Application Security Groups.

System Performance Measurements During Cell Repave

This table presents performance measurements taken during the Diego cell repave.

Note: These measurements indicate the peak cumulative values of the entire system (250 Diego cells, ~15,000 application instances, and 2 AZs.)

Use these measurements as a baseline for expected system load during Diego cell upgrade.

Measurement	Value	Metric Used
Cell CPU Consumption	36%	`bosh.healthmonitor.system.cpu.user`
Cell Memory Consumption	~50%	`bosh.healthmonitor.system.mem.percent`
Cell I/O Consumption (Read) During Normal Operations	43 Read I/O Operations per second	`aws.ebs.volume_read_ops`

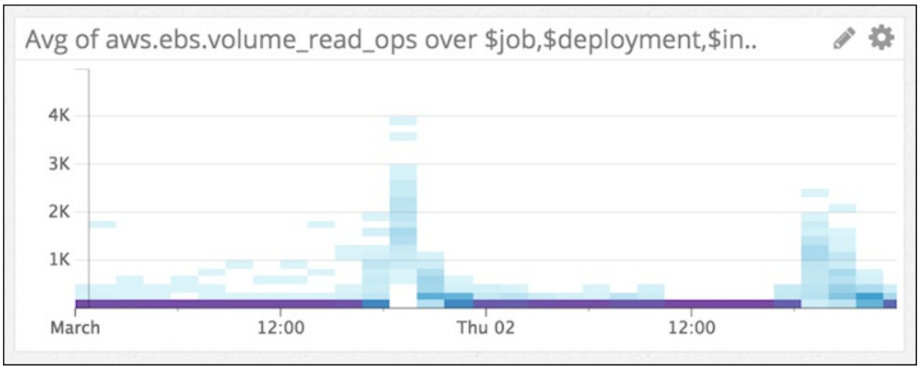
Cell I/O Consumption (Read) During Upgrade	1,943 Read I/O Operations per second	`aws.ebs.volume_read_ops`
Cell IO Consumption (Write) During Normal Operations	2,166 Write I/O Operations per second	`aws.ebs.volume_write_ops`
Cell IO Consumption (Write) During Upgrade	21,000 Write I/O Operations per second	`aws.ebs.volume_write_ops`
Cell Network Consumption (Network Out) During Normal Operations	~1.25 GB per minute	`aws.ec2.network_out`
Cell Network Consumption (Network Out) During Upgrade	~1.25 GB per minute (no significant change)	`aws.ec2.network_out`
Cell Network Consumption (Network In) During Normal Operations	2.11 GB per minute	`aws.ec2.network_in`
Cell Network Consumption (Network In) During Upgrade	16.75GB per minute	`aws.ec2.network_in`

Sample Performance Graphs

These DataDog graphs represent a timeline visualization of read and write operations during the repave event.

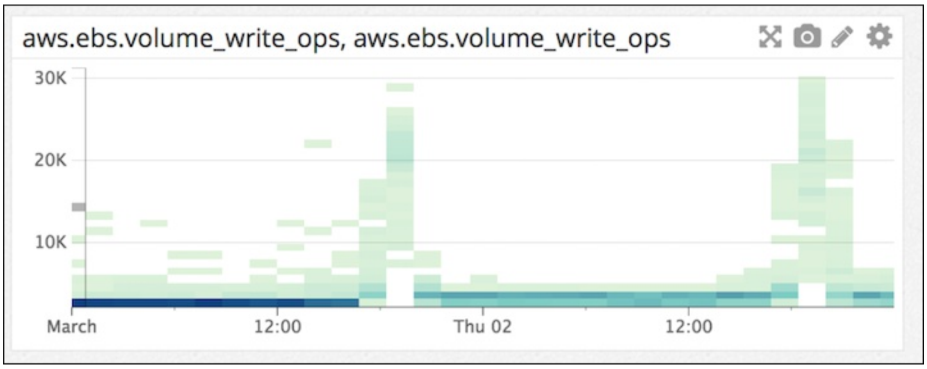
Read I/O Operations

The read I/O operations sample was taken over 115 VMs and represent the number of read operations over 300 seconds for a single Diego cell.



Write I/O Operations

The write I/O operations sample was taken over 115 VMs and represent the number of read I/O operations over 300 seconds for a single Diego cell.



Summary

During the repave process, 250 Diego cells were updated. The repave process took 6 hours overall or about 3 hours for each Availability Zone.

Pivotal

Upgrading PAS and Other Pivotal Cloud Foundry Products

Page last updated:

This topic describes how to upgrade to a point release of Pivotal Application Service (PAS) and other product tiles without upgrading Ops Manager. For example, use this topic to upgrade from PAS 1.9.0 to 1.9.1. You might need to perform this upgrade if a security update for PAS is released, or if new features are introduced in a point release of a product tile.

For PAS component and version information, see the PAS release notes.

Before You Upgrade to Point Releases

- You must have completed the Upgrading Pivotal Cloud Foundry procedure.
- Refer to the Product Compatibility Matrix before upgrading PAS.
- **Important**: Read the Known Issues sections of the products you plan on installing before starting. See Pivotal Cloud Foundry Release Notes for all available product release notes.

Upgrading PAS

Note: If you are using the Pivotal Network API, the latest product versions will automatically appear in your **Installation Dashboard**.

To upgrade PAS without upgrading Ops Manager, follow the procedure for installing PCF products:

1. Download the product file from Pivotal Network.
2. Import the product file to your Ops Manager **Installation Dashboard**.
3. Click the plus icon next to the uploaded product description to add this product to your staging area.
4. Click the newly added tile to review any configurable options.
5. Click **Apply Changes** to install the service.

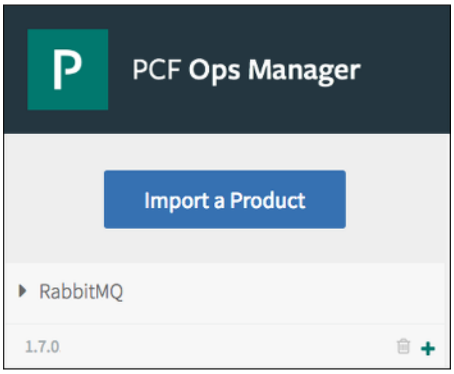
Upgrading PCF Products

Note: If you are using the Pivotal Network API, the latest product versions will automatically appear in your **Installation Dashboard**.

This section describes how to upgrade individual products like Single Sign-On for PCF, MySQL for PCF, RabbitMQ® for PCF, and Metrics for PCF in your Pivotal Cloud Foundry (PCF) deployment. Ensure you review the individual product upgrade procedure for each tile.

1. Browse to Pivotal Network and sign in.
2. Download the latest PCF release for the product or products you want to upgrade. Every product is tied to exactly one stemcell. Download the stemcell that matches your product and version.
3. Confirm that you have adequate disk space for your upgrades. You need at least 20 GB of free disk space to upgrade PCF Ops Manager and Pivotal Application Service (PAS). If you plan to upgrade other products, the amount of disk space required depends on how many tiles you plan to deploy to your upgraded PCF deployment.

   To check current persistent disk usage, select the **Ops Manager Director** tile from the **Installation Dashboard**. Select **Status** and review the value of the `PERS. DISK` column. If persistent disk usage is higher than 50%, select **Settings > Resource Config**, and increase your persistent disk space to handle the size of the resources. If you do not know how much disk space to allocate, set the value to at least `100 GB`.
4. Browse to the Pivotal Cloud Foundry Operations Manager web interface and click **Import a Product**.



5. Select the `.pivotal` file that you downloaded from Pivotal Network or received from your software distributor, then click **Open**. If the product is successfully added, it appears in the your product list. If the product you selected is not the latest version, the most up-to-date version will appear on your product list.

6. Click the plus icon next to the product description to add the product tile to the **Installation Dashboard**.

7. Repeat the import, upload, and upgrade steps for each product you downloaded.

8. If you are upgrading a product that uses a self-signed certificate from v1.1 to v1.2, you must configure the product to trust the self-signed certificate. Follow the steps below to configure a product to trust the self-signed certificate:

    a. Click the product tile.
    b. In the left-hand column, select the setting page containing the SSL certificate configuration. For example, for PAS, select the **HAProxy** page.
    c. Check the **Trust Self-Signed Certificates** box.
    d. Click **Save**.

9. Click **Apply changes**.

Reference Architectures

Introduction

A PCF reference architecture describes a proven approach for deploying Pivotal Cloud Foundry on a specific IaaS, such as AWS, Azure, GCP, or vSphere, that meets the following requirements:

- Secure
- Publicly-accessible
- Includes common PCF-managed services such as MySQL, RabbitMQ, and Spring Cloud Services
- Can host at least 100 app instances, or far more

These documents detail PCF reference architectures for different IaaSes to help you determine the best configuration for your PCF deployment.

Products Covered by the Reference Architectures

Pivotal has validated the following PCF products on its own deployments based on these reference architectures:

- Pivotal Cloud Foundry Ops Manager
- Pivotal Application Service (PAS)

Available Reference Architectures

- Pivotal Cloud Foundry on AWS
- Pivotal Cloud Foundry on Azure
- Pivotal Cloud Foundry on GCP
- Pivotal Cloud Foundry on OpenStack
- Pivotal Cloud Foundry on vSphere

Reference Architecture for Pivotal Cloud Foundry on AWS

Page last updated:

This guide presents a reference architecture for Pivotal Cloud Foundry (PCF) on Amazon Web Services (AWS). This architecture is valid for most production-grade PCF deployments using three availability zones (AZs).

See [AWS on PCF Requirements] for general requirements for running PCF and specific requirements for running PCF on AWS.

PCF Reference Architectures

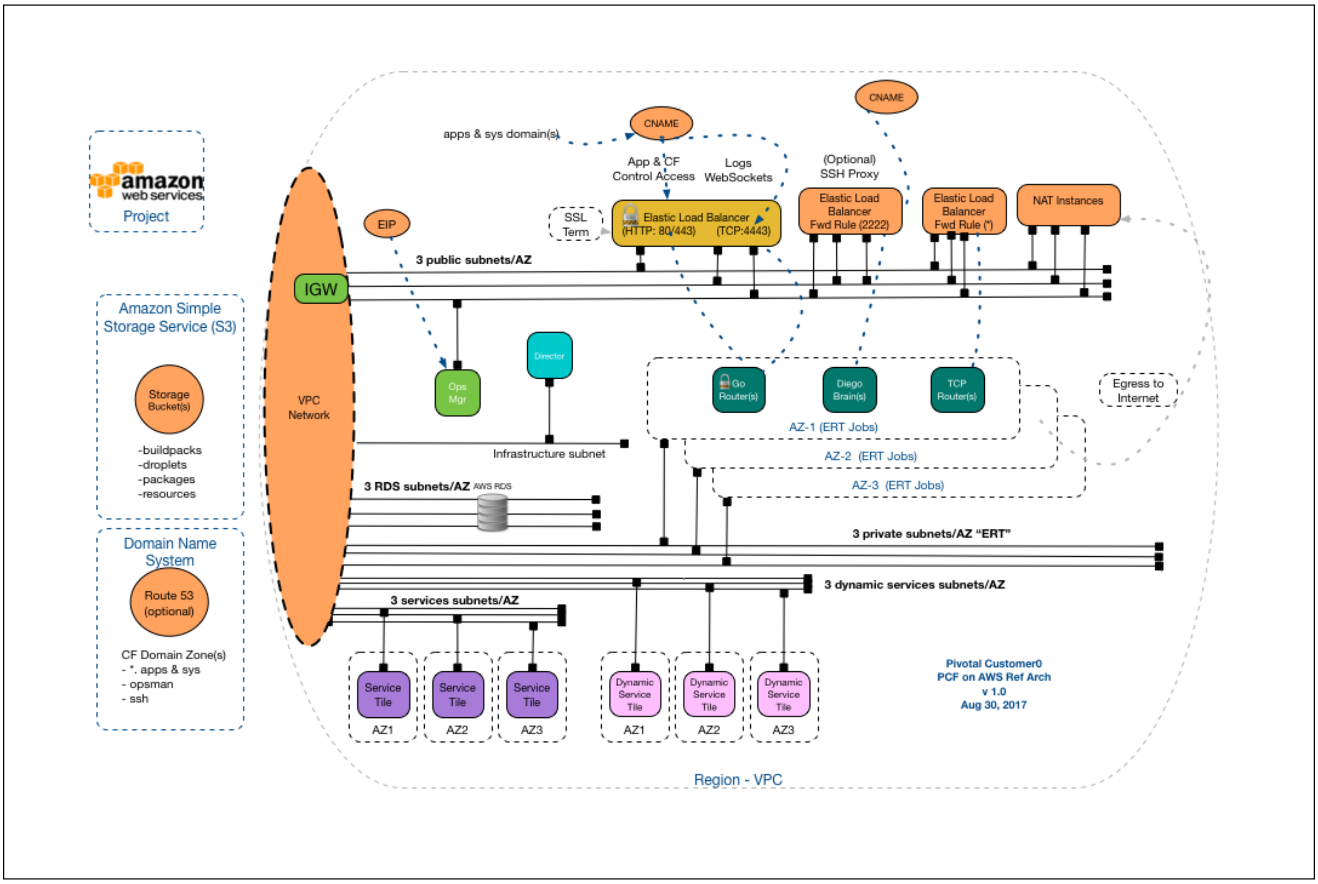
A PCF reference architecture describes a proven approach for deploying Pivotal Cloud Foundry on a specific IaaS, such as AWS, that meets the following requirements:

- Secure
- Publicly-accessible
- Includes common PCF-managed services such as MySQL, RabbitMQ, and Spring Cloud Services
- Can host at least 100 app instances, or far more

Pivotal provides reference architectures to help you determine the best configuration for your PCF deployment.

Base AWS Reference Architecture

The following diagram provides an overview of a reference architecture deployment of PCF on AWS using three AZs.



[View a larger version of this diagram].

Note: Each AWS subnet must reside entirely within one AZ. As a result, a multi-AZ deployment topology requires a subnet for each AZ.

Base Reference Architecture Components

The following table lists the components that are part of a base reference architecture deployment on AWS with three AZs.

Component	Reference Architecture Notes
Domains & DNS	CF Domain Zones and routes in use by the reference architecture include: • domains for *.apps and *.sys (required) • a route for Ops Manager (required) • a route for ssh access to app containers (optional) Using Route 53 to manage domains is optional.
Ops Manager	Deployed on one of the three public subnets and accessible by FQDN or through an optional jumpbox.
BOSH Director	Deployed on the infrastructure subnet.
Elastic Load Balancers - HTTP, HTTPS, and SSL	Required. Load balancer that handles incoming HTTP, HTTPS, and SSL traffic and forwards them to the Gorouters. Deployed on all three public subnets.
Elastic Load Balancers - SSH	Optional. Load balancer that provides SSH access to app containers. Deployed on all three public subnets, one per AZ.
Gorouters	Accessed through the HTTP, HTTPS, and SSL Elastic Load Balancers. Deployed on all three Pivotal Application Service (PAS) subnets, one per AZ.
Diego Brains	Required. However, the SSH container access functionality is optional and enabled through the SSH Elastic Load Balancers. Deployed on all three PAS subnets, one per AZ.
TCP Routers	Optional feature for TCP routing. Deployed on all three PAS subnets, one per AZ.
CF Database	Reference architecture uses AWS RDS. Deployed on all three RDS subnets, one per AZ.
Storage Buckets	Reference architecture uses 4 S3 buckets: buildpacks, droplets, packages, and resources.
Service Tiles	Deployed on all three service subnets, one per AZ.
Dynamic Services	Reserved for future use, dynamic services are deployed on their own subnet. Dynamic services are services autoprovisioned by BOSH based on a trigger, such as a request for that service. Pivotal recommends provisioning the multi-tenant dynamic services subnet as a /22 block.
Service User & Roles	One IAM role and one IAM user are recommended: the IAM role for Terraform, and the IAM user for Ops Manager and BOSH. Consult the following list: • Admin Role: Terraform will use this IAM role to provision required AWS resources as well as an IAM user. • IAM User: This IAM user with IAM security credentials (access key ID and secret access key) will be automatically provisioned with restrict access only to resources needed by PCF. See the AWS IAM Terraform script for more information.
EC2 Instance Quota	The default EC2 instance quota on a new AWS subscription only has around 20 EC2 instances, which is not enough to host a multi-AZ deployment. The recommended quota for EC2 instances is 100. AWS requires the instances quota tickets to include Primary Instance Types, which should be t2.micro.

Network Objects

The following table lists the network objects in this reference architecture.

Network Object	Notes	Estimated Number
External Public IPs	One per deployment, assigned to Ops Manager.	1
Virtual Private Network (VPC)	One per deployment. A PCF deployment exists within a single VPC and a single AWS region, but should distribute PCF jobs and instances across 3 AWS AZs to ensure a high degree of availability.	1
	The reference architecture requires the following subnets: • 1 x (/24) infrastructure (BOSH Director) subnet • 3 x (/24) public subnets (Ops Manager, Elastic Load Balancers, NAT instances), one per AZ	

<table>
<tr><td>Subnets</td><td>

3 x (/20) PAS subnets (Gorouters, Diego Cells, Cloud Controllers, etc.), one per AZ
3 x (/20) services subnets (RabbitMQ, MySQL, Spring Cloud Services, etc.), one per AZ
3 x (/22) dynamic services subnets (used for on-demand services), one per AZ
3 x (/24) RDS subnets (Cloud Controller DB, UAA DB, etc.), one per AZ.

For more information, see the Terraform subnets script.
</td><td>16</td></tr>
<tr><td>Route Tables</td><td>
This reference architecture requires 4 route tables: one for the public subnet, and one each for all 3 private subnets across 3 AZs. Consult the following list:

PublicSubnetRouteTable: This routing table enables the ingress/egress routes from/to Internet through the Internet gateway for Ops Manager and the NAT Gateway.
PrivateSubnetRouteTable: This routing table enables the egress routing to the Internet through the NAT Gateway for the BOSH Director and PAS.

For more information, see the Terraform script that creates the route tables and the script that performs the route table association.

Note: If an EC2 instance sits on a subnet with an Internet gateway attached as well as a public IP address, it is accessible from the Internet through the public IP address; for example, Ops Manager. PAS needs Internet access due to the access needs of using an S3 bucket as a blobstore.
</td><td>4</td></tr>
<tr><td>Security Groups</td><td>
The reference architecture requires 5 Security Groups. For more information, see the Terraform Security Group rules script. The following table describes the Security Group ingress rules:

Note: The extra port of 4443 with the Elastic Load Balancer is due to the limitation that the Elastic Load Balancer does not support WebSocket connections on HTTP/HTTPS.

<table>
<tr><th>Security Group</th><th>Port</th><th>From CIDR</th><th>Protocol</th><th>Description</th></tr>
<tr><td>OpsMgrSG</td><td>22</td><td>0.0.0.0/0</td><td>TCP</td><td>Ops Manager SSH access</td></tr>
<tr><td>OpsMgrSG</td><td>443</td><td>0.0.0.0/0</td><td>TCP</td><td>Ops Manager HTTP access</td></tr>
<tr><td>VmsSG</td><td>ALL</td><td>VPC_CIDR</td><td>ALL</td><td>Open up connections among BOSH-deployed VMs</td></tr>
<tr><td>MysqlSG</td><td>3306</td><td>VPC_CIDR</td><td>TCP</td><td>Enable network access to RDS</td></tr>
<tr><td>ElbSG</td><td>80</td><td>0.0.0.0/0</td><td>TCP</td><td>HTTP to PAS</td></tr>
<tr><td>ElbSG</td><td>443</td><td>0.0.0.0/0</td><td>TCP</td><td>HTTPS to PAS</td></tr>
<tr><td>ElbSG</td><td>4443</td><td>0.0.0.0/0</td><td>TCP</td><td>WebSocket connection to Loggregator endpoint</td></tr>
<tr><td>SshElbSG</td><td>2222</td><td>0.0.0.0/0</td><td>TCP</td><td>SSH connection to containers</td></tr>
</table>
</td><td>5</td></tr>
<tr><td>Load Balancers</td><td>
PCF on AWS requires the Elastic Load Balancer, which can be configured with multiple listeners to forward HTTP/HTTPS/TCP traffic. Two Elastic Load Balancers are recommended: one to forward the traffic to the Gorouters, <code>PcfElb</code>, the other to forward the traffic to the Diego Brain SSH proxy, <code>PcfSshElb</code>. For more information, see the Terraform load balancers script.

The following table describes the required listeners for each load balancer:

<table>
<tr><th>ELB</th><th>Instance/Port</th><th>LB Port</th><th>Protocol</th><th>Description</th></tr>
<tr><td>PcfElb</td><td>gorouter/80</td><td>80</td><td>HTTP</td><td>Forward traffic to Gorouters</td></tr>
<tr><td>PcfElb</td><td>gorouter/80</td><td>443</td><td>HTTPS</td><td>SSL termination and forward traffic to Gorouters</td></tr>
<tr><td>PcfElb</td><td>gorouter/80</td><td>4443</td><td>SSL</td><td>SSL termination and forward traffic to Gorouters</td></tr>
<tr><td>PcfSshElb</td><td>diego-brain/2222</td><td>2222</td><td>TCP</td><td>Forward traffic to Diego Brain for container SSH connections</td></tr>
</table>
Each ELB binds with a health check to check the health of the back-end instances:

<code>PcfElb</code> checks the health on Gorouter port 80 with TCP
<code>PcfSshElb</code> checks the health on Diego Brain port 2222 with TCP

</td><td>2</td></tr>
<tr><td>Jumpbox</td><td>Optional. Provides a way of accessing different network components. For example, you can configure it with your own permissions and then set it up to access to Pivotal Network to download tiles. Using a jumpbox is particularly useful in IaaSes where Ops Manager does not have a public IP address. In these cases, you can SSH into Ops Manager or any other component through the jumpbox.</td><td>1</td></tr>
</table>

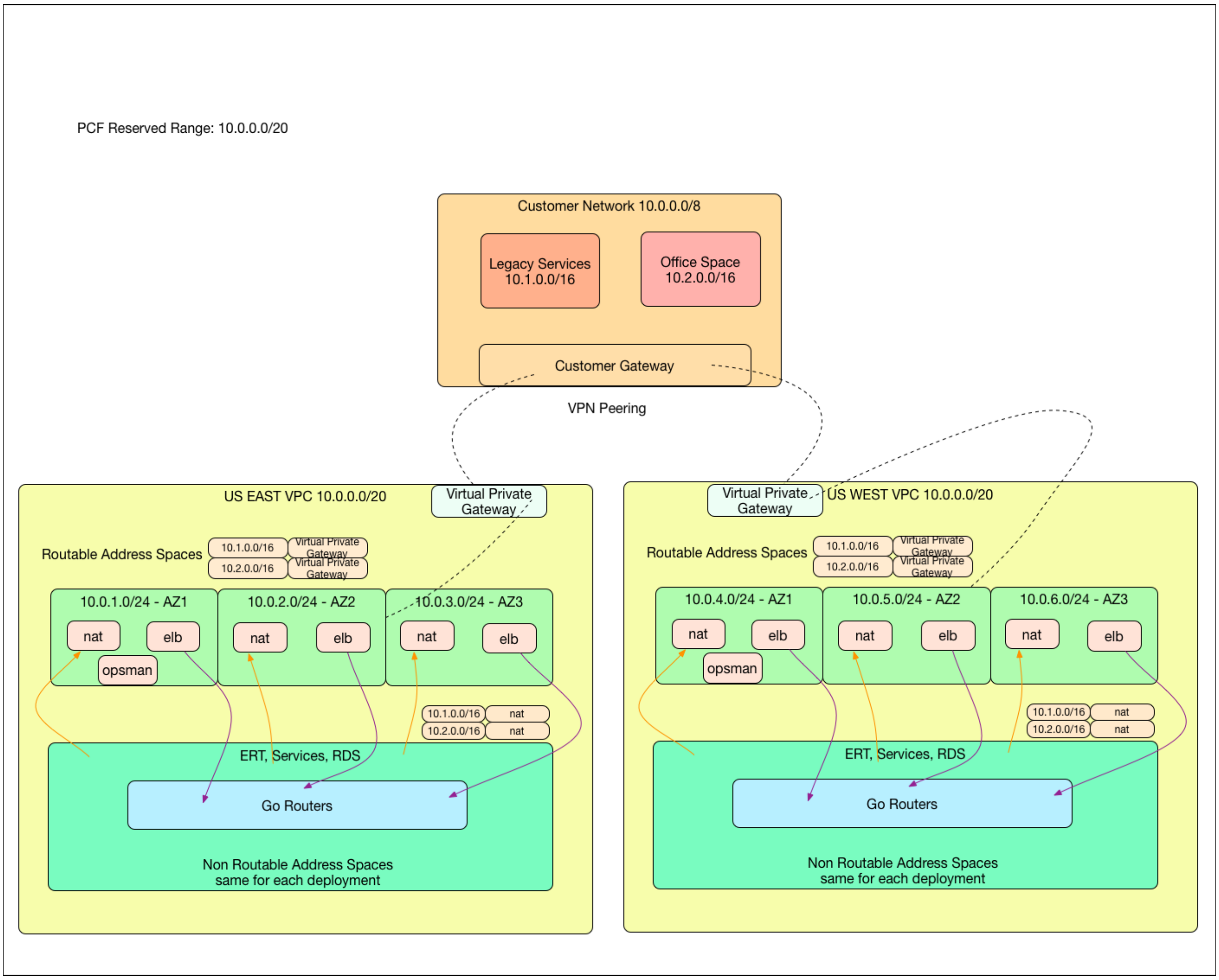
Integrate PCF with Customer Data Center through VPN

At times, applications on PCF need to access on-premise data. The connection between an AWS VPC and an on-premise datacenter is made throughVPN peering. When employing non-VPN peering, there are several points to consider:

1. Assign routable IP addresses with the following in mind:
    - It may not be realistic to request multiple routable /22 address spaces, due to IP exhaustion.
    - Using different VPC address spaces can cause snowflakes deployments and present difficulties in automation.
    - Only make the load balancer, NAT devices, and Ops Manager routable.
    - PCF components can route egress through a NAT instance. As a result, operators do not need to assign routable IP addresses to PCF components.

2. Inbound traffic from the datacenter should come through an internal load balancer.

3. Outbound traffic to the datacenter should go through AWS NAT instances.



View a larger version of this diagram.

Reference Architecture for Pivotal Cloud Foundry on Azure

Page last updated:

This guide presents a reference architecture for Pivotal Cloud Foundry (PCF) on Azure.

Azure does not provide resources in a way that translates directly to PCF availability zones. Instead, Azure provides high availability through fault domains and availability sets.

All reference architectures described in this topic are validated for production-grade PCF deployments using fault domains and availability sets that include multiple job instances.

See Azure on PCF Requirements for general requirements for running PCF and specific requirements for running PCF on Azure.

PCF Reference Architectures

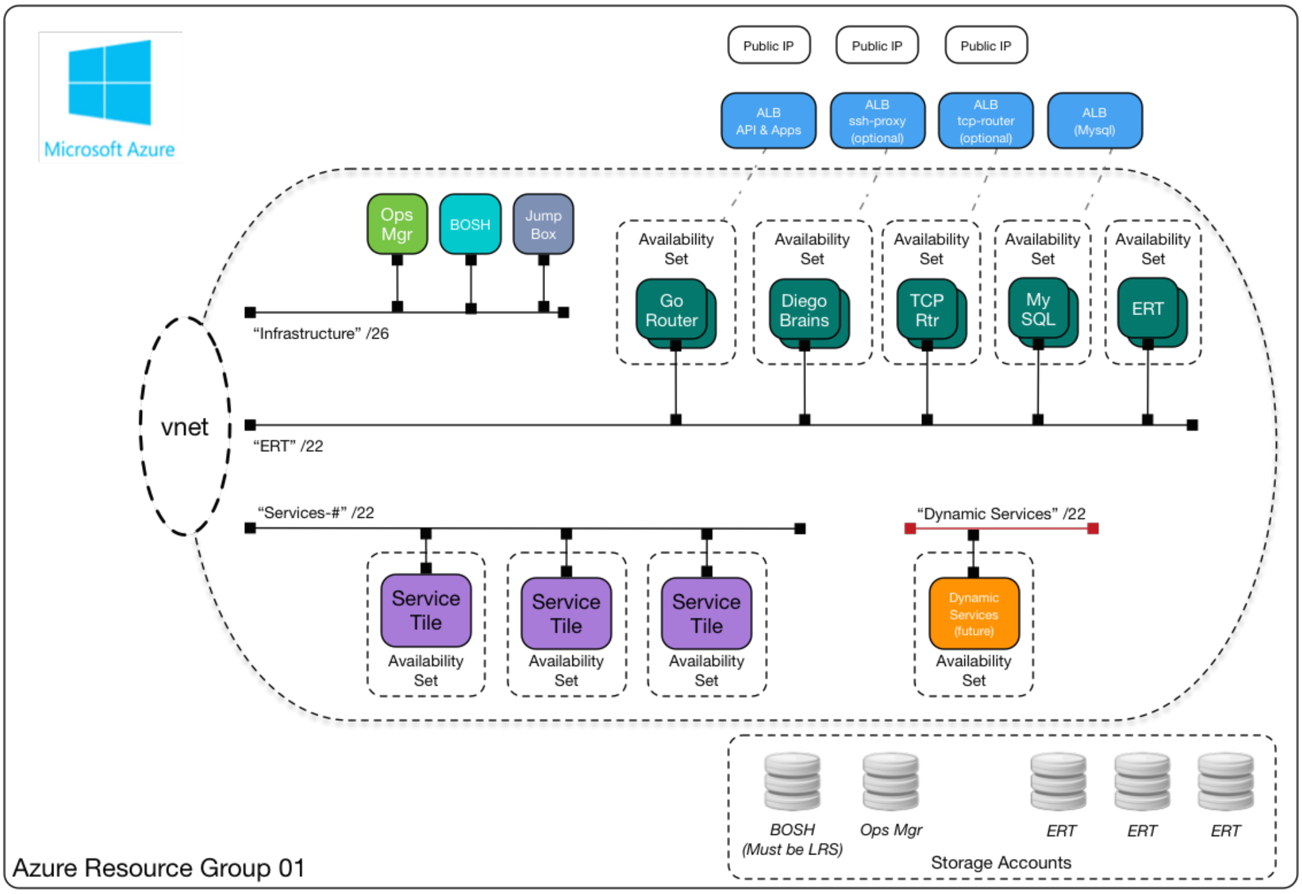
A PCF reference architecture describes a proven approach for deploying Pivotal Cloud Foundry on a specific IaaS, such as Azure, that meets the following requirements:

- Secure
- Publicly-accessible
- Includes common PCF-managed services such as MySQL, RabbitMQ, and Spring Cloud Services
- Can host at least 100 app instances, or far more

Pivotal provides reference architectures to help you determine the best configuration for your PCF deployment.

Base Azure Reference Architecture

The following diagram provides an overview of a reference architecture deployment of PCF on Azure.



View a larger version of this diagram.

Base Reference Architecture Components

The following table lists the components that are part of a base reference architecture deployment on Azure using a single resource group.

Component	Reference Architecture Notes
Domains and DNS	CF Domain Zones and routes in use by the reference architecture include: • domains for *.apps and *.system (required), • a route for Ops Manager (required), • a route for Doppler (required), • a route for Loggregator (required), • a route for ssh access to app containers (optional), • and a route for TCP routing to apps (optional).
Ops Manager	Deployed on the infrastructure subnet and accessible by FQDN or through an optional jumpbox.
BOSH	Deployed on the infrastructure subnet.
Azure Load Balancer - API and Apps	Required. Load balancer that handles incoming API and apps requests and forwards them to the Gorouters.
Azure Load Balancer - ssh-proxy	Optional. Load balancer that provides SSH access to app containers.
Azure Load Balancer - tcp-router	Optional. Load balancer that handles TCP routing requests for apps.
Azure Load Balancer - MySQL	Required to provide high availability for MySQL backend to Pivotal Application Service (PAS).
Gorouters	Accessed through the API and Apps load balancer. Deployed on the PAS subnet, one job per Azure availability set.
Diego Brains	This component is required, however the SSH container access functionality is optional and enabled through the SSH Proxy load balancer. Deployed on the PAS subnet, one job per Azure availability set.
TCP Routers	Optional feature for TCP routing. Deployed on the PAS subnet, one job per availability zone.
MySQL	Reference architecture uses internal MySQL provided with PCF. Deployed on the PAS subnet, one job per Azure availability set.
PAS	Required. Deployed on the PAS subnet, one job per Azure availability set.
Storage Accounts	PCF on Azure requires 5 standard storage accounts: BOSH, Ops Manager, and three PAS storage accounts. Each account comes with a set amount of disk. Reference architecture recommends using 5 storage accounts because Azure Storage Accounts have an IOPs limit of approximately 20k per account, which generally relates to a BOSH JOB/VM limit of approximately 20 VMs each.
Service Tiles	Deployed on the PCF managed services subnet. Each service tile is deployed to an availability set.
Dynamic Services	Reserved for future use, dynamic services are deployed on their own subnet. Dynamic services are services autoprovisioned by BOSH based on a trigger, such as a request for that service. Pivotal recommends provisioning the multi-tenant dynamic services subnet as a /22 block.

Alternative Network Layouts for Azure

This section describes the possible network layouts for PCF deployments as covered by the reference architecture of PCF on Azure.

At a high level, there are currently two possible ways of deploying PCF as described by the reference architecture:

1. Single resource group, or

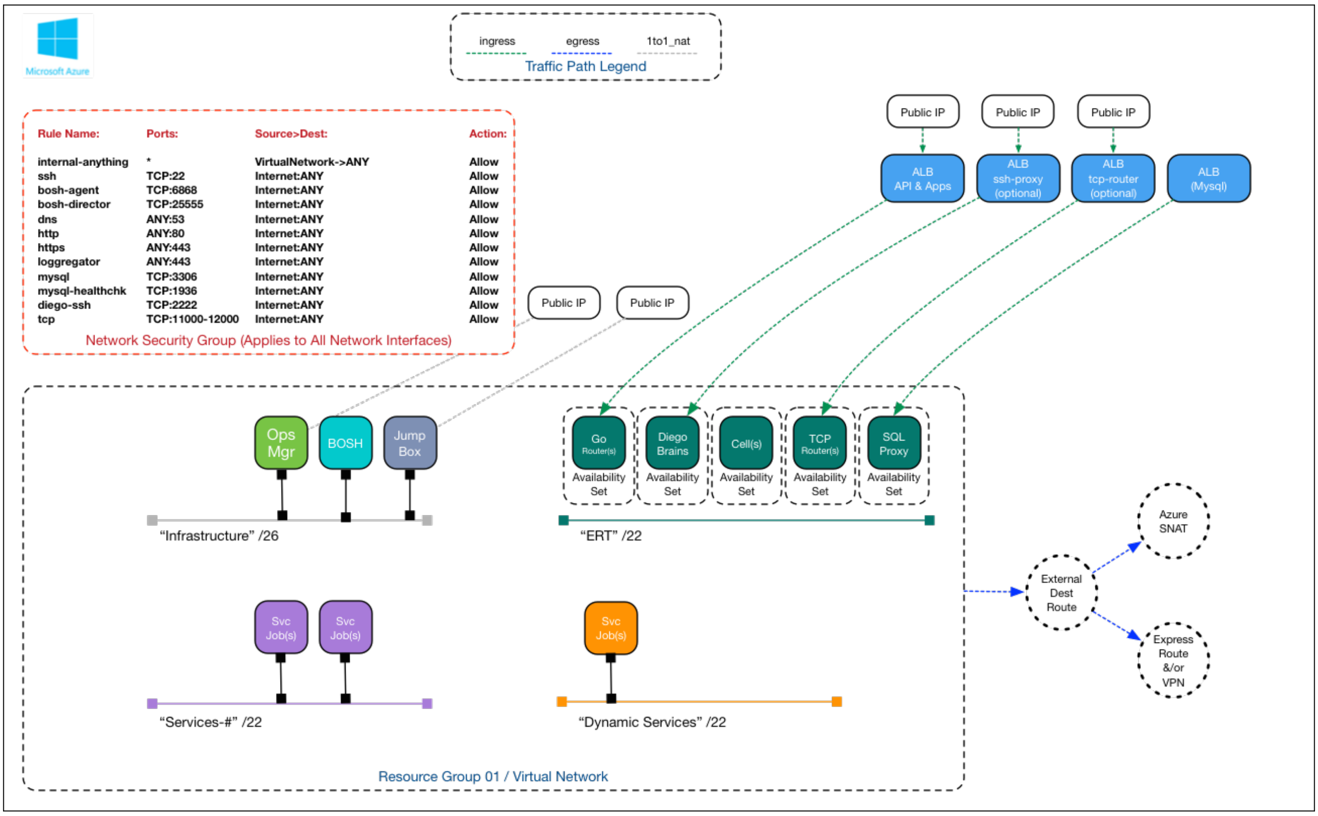
2. Multiple resource groups.

The first scenario is outlined in the [Installing PCF on Azure]. It models a single PCF deployment in a single Azure Resource Group.

If you require multiple resource groups, refer to the Multiple Resource Group deployment section of this topic.

Network Layout

This diagram illustrates the network topology of the base reference architecture for PCF on Azure. In this deployment, you expose only a minimal number of public IP addresses and deploy only one resource group.



View a larger version of this diagram.

Network Objects

The following table lists the network objects in PCF on Azure reference architecture.

Network Object	Notes	Estimated Number
External Public IP addresses	Use 1. global IP address for apps and system access 2. Ops Manager or optional jumpbox. Optionally, you can use a public IP address for the ssh-proxy and tcp-router load balancers.	1-4
Virtual Network	One per deployment. Azure virtual network objects allow multiple subnets with multiple CIDRs, so a typical deployment of PCF will likely only ever require one Azure Virtual Network object.	1
Subnets	Separate subnets for 1. infrastructure (Ops Manager, Ops Manager Director, Jumpbox), 2. PAS, 3. services,	4

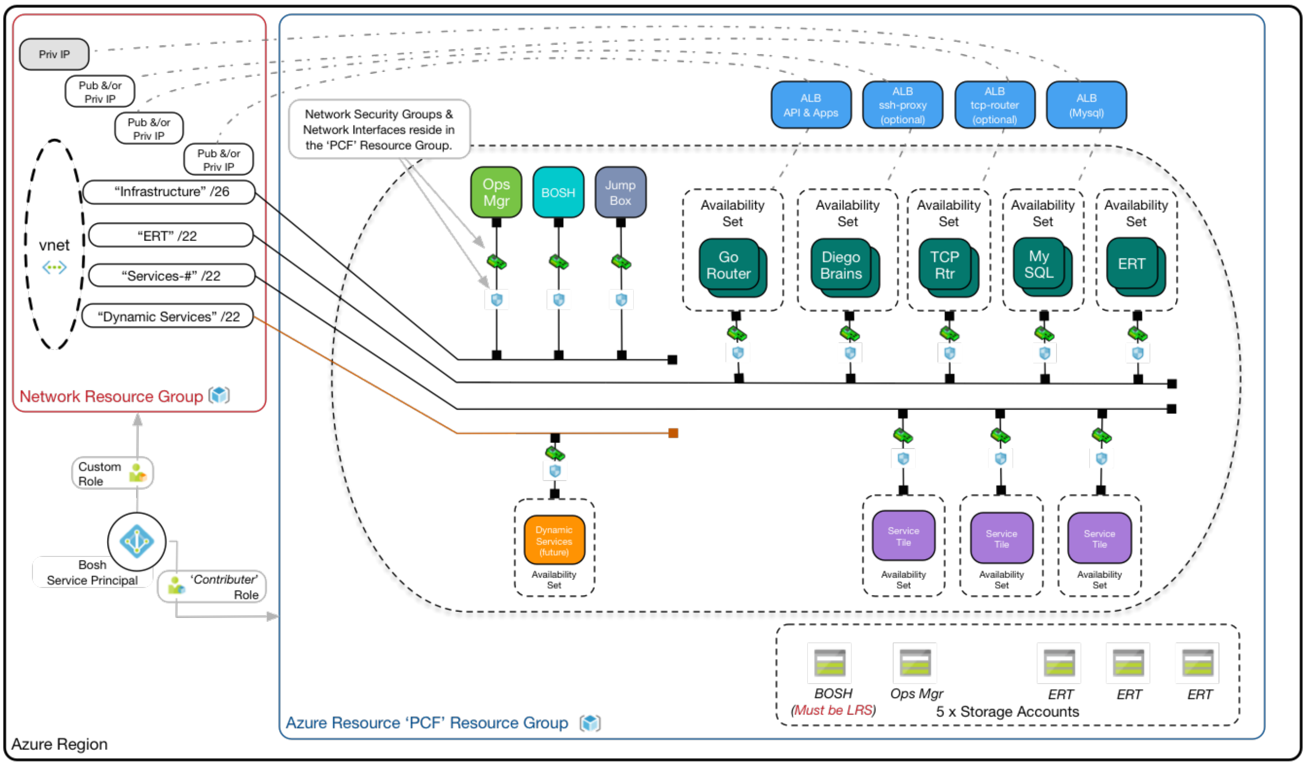
	4. and dynamic services. Using separate subnets allows you to configure different firewall rules due to your needs.	
Routes	Routes are typically created by Azure dynamically when subnets are created, but you may need to create additional routes to force outbound communication to dedicated SNAT nodes. These objects are required to deploy PCF without public IP addresses.	3+
Firewall Rules	Azure firewall rules are collected into a Network Security Group (NSG) and bound to a Virtual Network object and can be created to use IP ranges, subnets, or instance tags to match for source and destination fields in a rule. One NSG can be used for all firewall rules.	12
Load Balancers	Used to handle requests to Gorouters and infrastructure components. Azure uses 1 or more load balancers. The API and Apps load balancer is required. The TCP Router load balancer used for TCP routing feature and the SSH load balancer that allows SSH access to Diego apps are both optional. In addition, you can use a MySQL load balancer to provide high availability to MySQL. This is also optional.	1-4
Jumpbox	Optional. Provides a way of accessing different network components. For example, you can configure it with your own permissions and then set it up to access to Pivotal Network to download tiles. Using a jumpbox is particularly useful in IaaSes where Ops Manager does not have a public IP address. In these cases, you can SSH into Ops Manager or any other component through the jumpbox.	1

Multiple Resource Group Deployment

This diagram illustrates the case where you want to use additional resource groups in your PCF deployment on Azure.

Shared network resources may already exist in an Azure subscription. In this type of deployment, using multiple resource groups allows you to reuse existing resources instead of provisioning new ones.

To use multiple resource groups, you need to provide the BOSH Service Principal with access to the existing network resources.



View a larger version of this diagram.

Multiple Resource Groups Deployment Notes

To deploy PCF on Azure with multiple resource groups, you can define custom roles to grant resource group access to your BOSH Service Principal. For example, you might develop the following:

- Dedicated `Network` Resource Group, limits BOSH Service Principal so that it does not have admin access to network objects.
- Custom Role for BOSH Service Principal, applied to `Network` Resource Group, limits the BOSH Service Principal to minimum read-only access.

```
{
"Name": "PCF Network Read Only",
"IsCustom": true,
"Description": "MVP PCF Read Network Resgroup",
"Actions": [
"Microsoft.Network/networkSecurityGroups/read",
"Microsoft.Network/networkSecurityGroups/join/action",
"Microsoft.Network/publicIPAddresses/read",<-- Only Required if Using Public IPs
"Microsoft.Network/publicIPAddresses/join/action", <-- Only Required if Using Public IPs
"Microsoft.Network/loadBalancers/read",
"Microsoft.Network/virtualNetworks/subnets/read",
"Microsoft.Network/virtualNetworks/subnets/join/action",
"Microsoft.Network/virtualNetworks/read"
],
"NotActions": [],
"AssignableScopes": ["/subscriptions/[YOUR_SUBSCRIPTION_ID]"]
}
```

- Custom Role for BOSH Service Principal, applied to Subscription, allowing the Operator to deploy PCF components

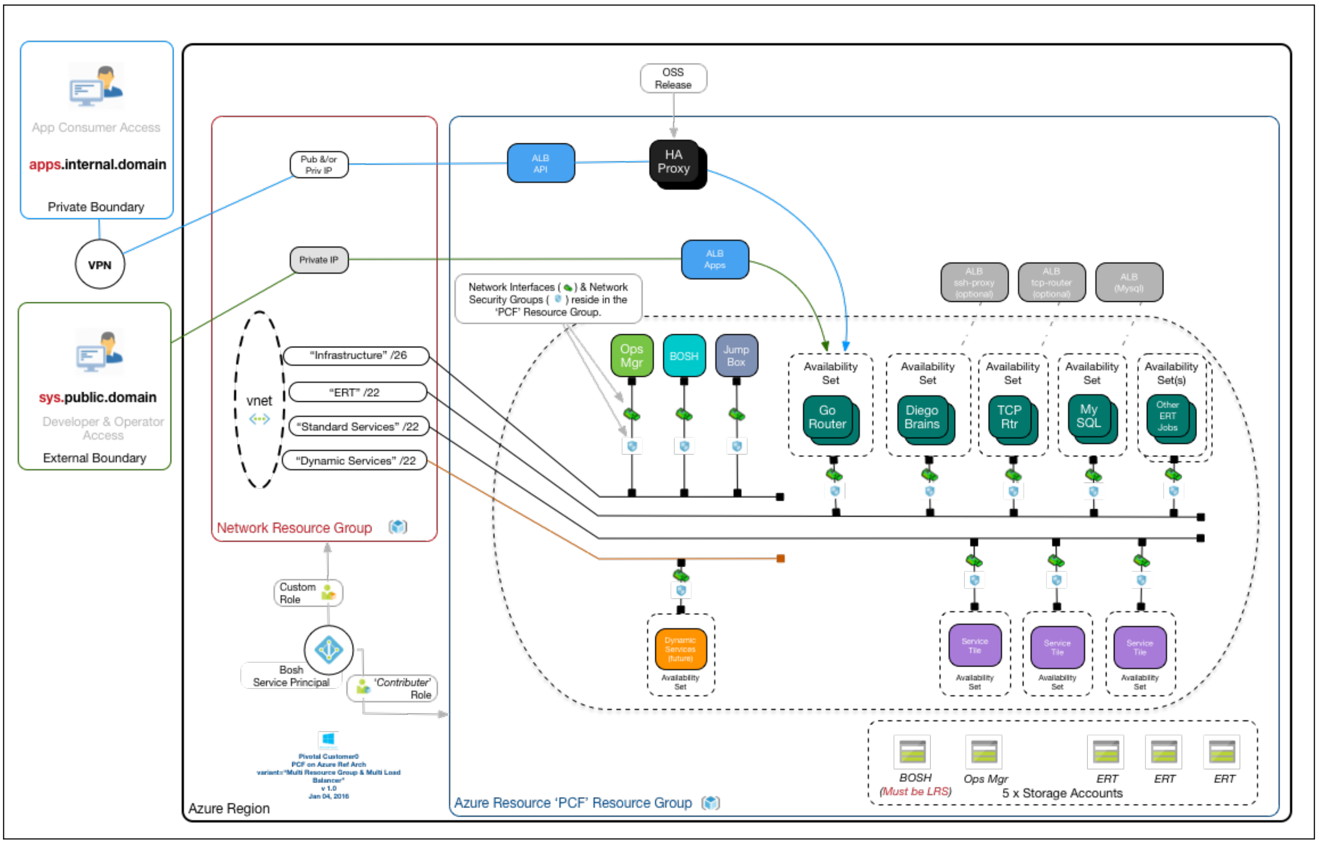
```
{
"Name": "PCF Deploy Min Perms",
"IsCustom": true,
"Description": "MVP PCF Terraform Perms",
"Actions": [
 "Microsoft.Compute/register/action"
],
"NotActions": [],
"AssignableScopes": ["/subscriptions/[YOUR_SUBSCRIPTION_ID]"]
}
```

Multiple Resource Group and Multiple Load Balancer Deployment

This diagram illustrates the case where you want to deploy multiple load balancers with your multiple resource group deployment.

The key design points of this deployment are the following:

- Two Azure load balancer (ALBs) to the Gorouter exist. The first ALB is for API access, which utilizes a private IP address. The system domain should resolve to this ALB. The second ALB is for application access, which can either use a public or private IP address. The apps domain should resolve to this ALB.
- Azure does not allow the Gorouters to be members of more than one ALB member pool for the same ports, for example `80` and `443`. This restriction requires an additional reverse proxy to front the Gorouters to allow hem to expose traffic on these ports for the system domain.



[View a larger version of this diagram](#).

Reference Architecture for Pivotal Cloud Foundry on GCP

Page last updated:

This guide presents a reference architecture for Pivotal Cloud Foundry (PCF) on Google Cloud Platform (GCP). This document also outlines multiple networking solutions. All these architectures are validated for production-grade PCF deployments using multiple (3+) AZs.

See GCP on PCF Requirements for general requirements for running PCF and specific requirements for running PCF on GCP.

PCF Reference Architectures

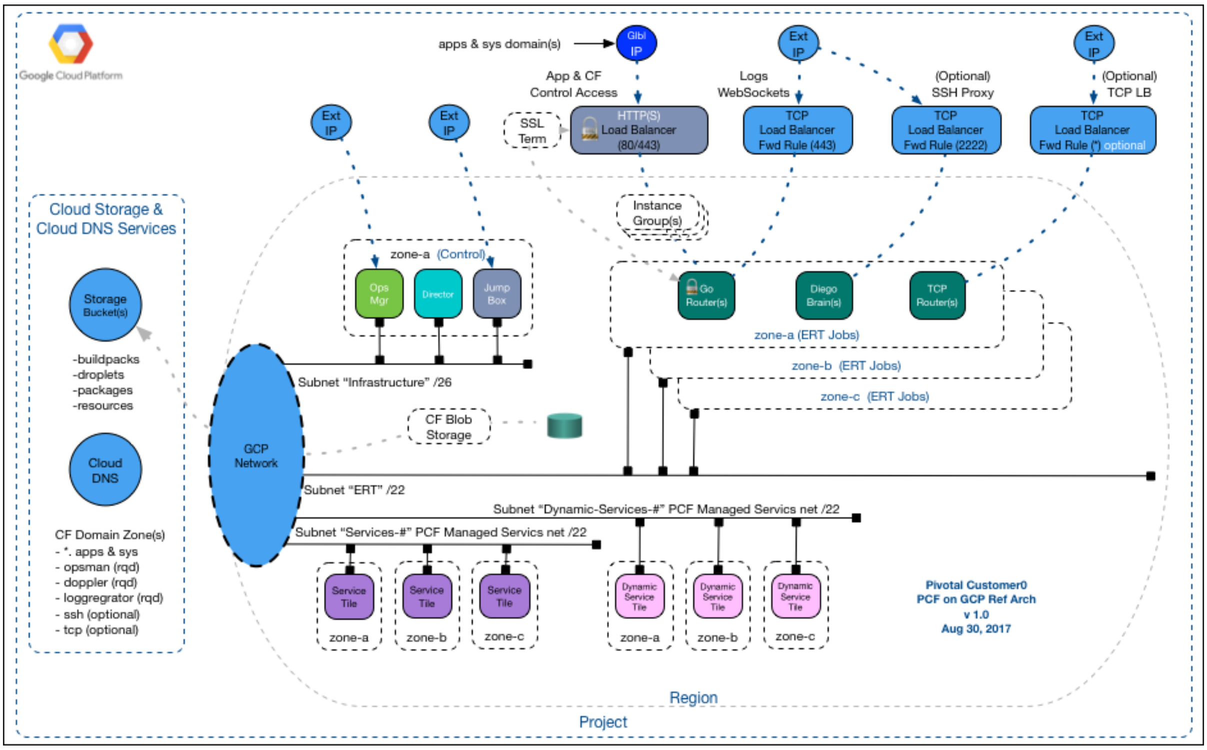
A PCF reference architecture describes a proven approach for deploying Pivotal Cloud Foundry on a specific IaaS, such as GC, that meets the following requirements:

- Secure
- Publicly-accessible
- Includes common PCF-managed services such as MySQL, RabbitMQ, and Spring Cloud Services
- Can host at least 100 app instances, or far more

Pivotal provides reference architectures to help you determine the best configuration for your PCF deployment.

Base GCP Reference Architecture

The following diagram provides an overview of a reference architecture deployment of PCF on GCP.



View a larger version of this diagram.

Base Reference Architecture Components

The following table lists the components that are part of a reference architecture deployment with three availability zones.

Component	Reference Architecture Notes

Component	Reference Architecture Notes
Domains & DNS	CF Domain Zones and routes in use by the reference architecture include: domains for *.apps and *.system (required), a route for Ops Manager (required), a route for doppler (required), a route for Loggregator (required), a route for ssh access to app containers (optional) and a route for TCP routing to apps (optional). Reference architecture uses GCP Cloud DNS as the DNS provider.
Ops Manager	Deployed on the infrastructure subnet and accessible by FQDN or through an optional jumpbox.
BOSH Director	Deployed on the infrastructure subnet.
Gorouters	Accessed through the HTTP and TCP WebSockets load balancers. Deployed on the Pivotal Application Service (PAS) subnet, one job per availability zone.
Diego Brains	Required. However, the SSH container access functionality is optional and enabled through the SSH Proxy load balancer. Deployed on the PAS subnet, one job per availability zone.
TCP Routers	Optional feature for TCP routing. Deployed on the PAS subnet, one job per availability zone.
CF Database	Reference architecture uses GCP Cloud SQL rather than internal databases. Configure your database with a strong password and limit access only to components that require database access.
CF Blob Storage and Buckets	For buildpacks, droplets, packages and resources. Reference architecture uses Google Cloud Storage rather than internal file storage.
Services	Deployed on the PCF managed services subnet. Each service is deployed to each availability zone.
Dynamic Services	Reserved for future use, dynamic services are deployed on their own subnet. Dynamic services are services autoprovisioned by BOSH based on a trigger, such as a request for that service. Pivotal recommends provisioning the multi-tenant dynamic services subnet as a /22 block.

Alternative GCP Network Layouts for PCF

This section describes the possible network layouts for PCF deployments as covered by the reference architecture of PCF on GCP.

At a high level, there are currently two possible ways of granting public Internet access to PCF as described by the reference architecture:

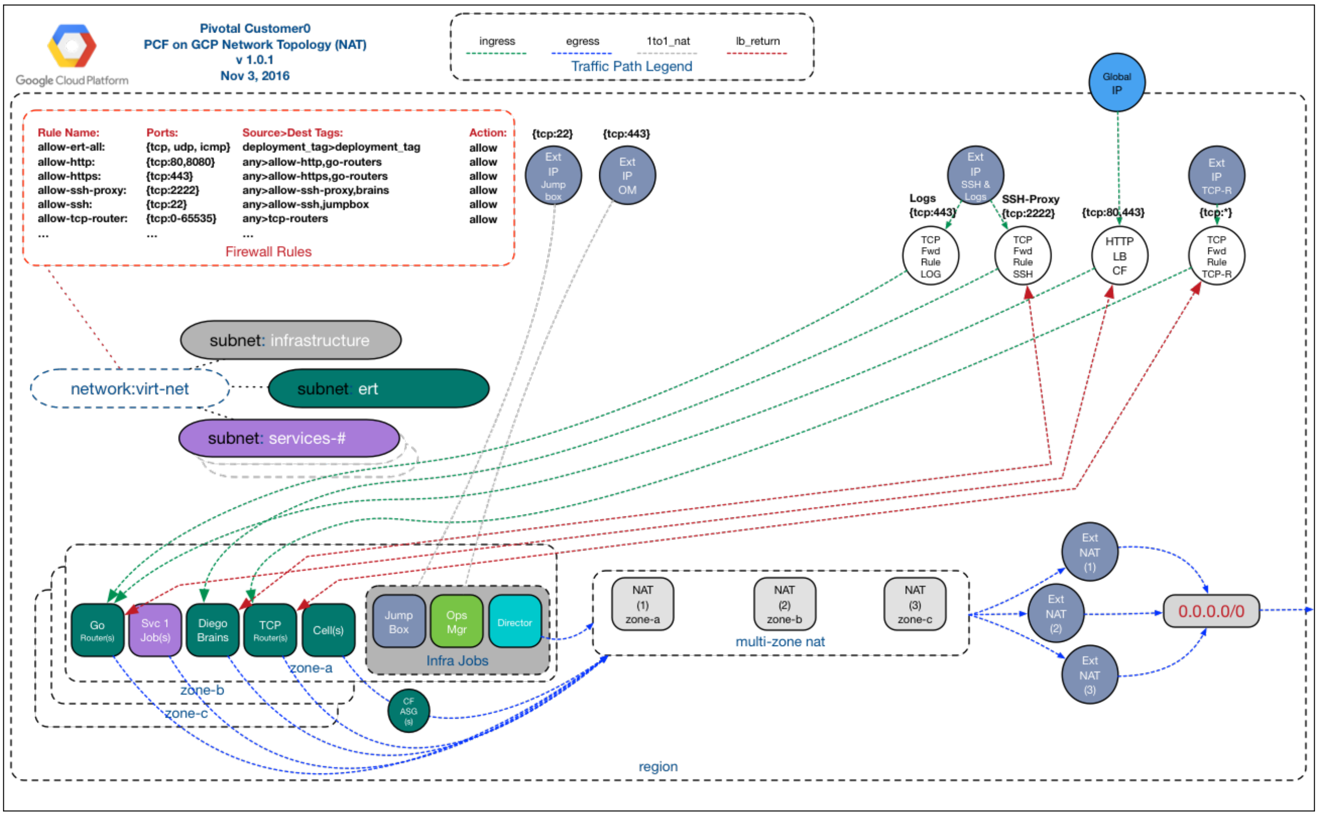
1. NATs provide public access to PCF internals, or
2. Every PCF VM receives its own public IP address (no NAT).

The latter scenario is currently outlined in the existing installation documentation for GCP deployments of PCF. Providing each PCF VM with a public IP address is the most recommended architecture, because of increased latency due to NATs as well as extra maintenance required for NAT instances that cannot be deployed with BOSH.

However, if you require NATs, you may refer to the following section.

NAT-based Solution

This diagram illustrates the case where you want to expose only a minimal number of public IP addresses.



[View a larger version of this diagram](#).

Public IP addresses Solution

If you prefer not to use a NAT solution, you can configure PCF on GCP to assign public IP addresses for all components. This type of deployment may be more performant since most of the network traffic between Cloud Foundry components are routed through the front end load balancer and the Gorouter.

Network Objects

The following table lists the network objects expected for each type of reference architecture deployment with three availability zones (assumes you are using NATs).

Network Object	Notes	Minimum Number: NAT-based	Minimum Number: Public IP Addresses
External IPs	For a NAT solution, use global IP address for apps and system access, and Ops Manager or optional jumpbox	2	30+
NATs	One NAT per availability zone.	3	0
Network	One per deployment. GCP Network objects allow multiple subnets with multiple CIDRs, so a typical deployment of PCF will likely only ever require one GCP Network object	1	1
Subnets	Separate subnets for infrastructure (Ops Manager, Ops Manager Director, Jumpbox), PAS, and services. Using separate subnets allows you to configure different firewall rules due to your needs.	3	3
Routes	Routes are typically created by GCP dynamically when subnets are created, but you may need to create additional routes to force outbound communication to dedicated SNAT nodes. These objects are required to deploy PCF without public IP addresses.	3+	3
Firewall Rules	GCP firewall rules are bound to a Network object and can be created to use IP ranges, subnets, or instance tags to match for source & destination fields in a rule. The preferred method use in the reference architecture deployment is instance tags.	6+	6+
Load balancers	Used to handle requests to Gorouters and infrastructure components. GCP uses two or more load balancers. The HTTP load balancer and TCP WebSockets load balancer are both required. The TCP Router load balancer used for TCP routing feature and the SSH load balancer that allows SSH access to Diego apps are both optional. The	2+	2+

Network Object	Notes	Minimum Number: NAT-based (1)	Minimum Number: Public IP Addresses (1)
	HTTP load balancer provides SSL termination.		
Jumpbox	Optional. Provides a way of accessing different network components. For example, you can configure it with your own permissions and then set it up to access to Pivotal Network to download tiles. Using a jumpbox is particularly useful in IaaSes where Ops Manager does not have a public IP address. In these cases, you can SSH into Ops Manager or any other component through the jumpbox.		

Network Communication in GCP Deployments

This section provides more background on the reasons behind certain network configuration decisions, specifically for the Gorouter.

Load Balancer to Gorouter Communications and TLS Termination

In a PCF on GCP deployment, the Gorouter receives two types of traffic:

1. Unencrypted HTTP traffic on port 80 that is decrypted by the HTTP(S) load balancer.

2. Encrypted secure web socket traffic on port 443 that is passed through the TCP WebSockets load balancer.

TLS is terminated for HTTPS traffic on the HTTP load balancer and is terminated for WebSockets (WSS) traffic on the Gorouter.

PCF deployments on GCP use two load balancers to handle Gorouter traffic because HTTP load balancers currently do not support WebSockets.

ICMP

GCP routers do not respond ICMP; therefore, Pivotal recommends disabling ICMP checks in Ops Manager Director network configuration.

Reference Architecture for Pivotal Cloud Foundry on OpenStack

Page last updated:

This guide presents a reference architecture for Pivotal Cloud Foundry (PCF) on OpenStack. This architecture is valid for most production-grade PCF deployments in a single project using three availability zones (AZs).

See OpenStack on PCF Requirements for general requirements for running PCF and specific requirements for running PCF on OpenStack.

PCF Reference Architectures

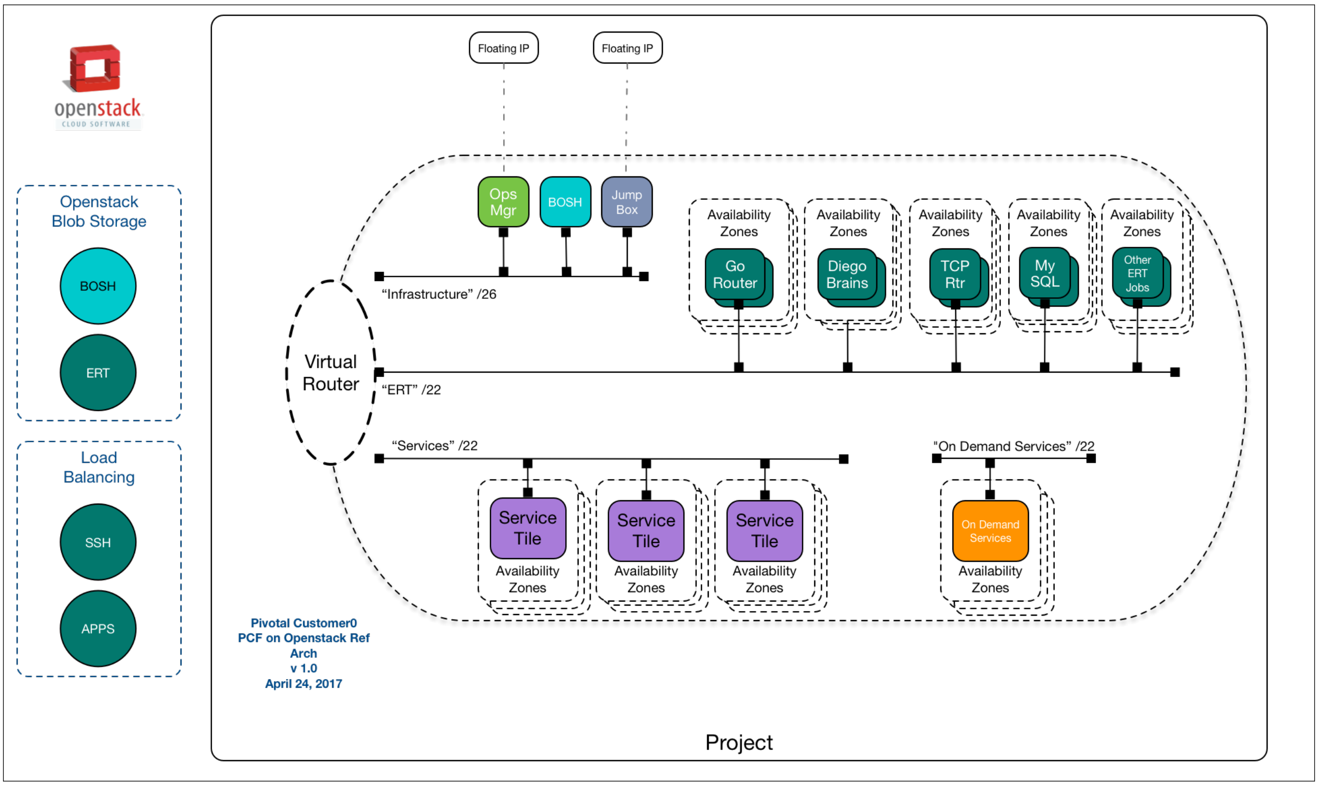
A PCF reference architecture describes a proven approach for deploying Pivotal Cloud Foundry on a specific IaaS, such as OpenStack, that meets the following requirements:

- Secure
- Publicly-accessible
- Includes common PCF-managed services such as MySQL, RabbitMQ, and Spring Cloud Services
- Can host at least 100 app instances, or far more

Pivotal provides reference architectures to help you determine the best configuration for your PCF deployment.

Base OpenStack Reference Architecture

The following diagram provides an overview of a reference architecture deployment of PCF on OpenStack using three AZs.



View a larger version of this diagram.

Base Reference Architecture Components

The following table lists the components that are part of a base reference architecture deployment on OpenStack with three AZs.

Component	Reference Architecture Notes
Domains & DNS	CF Domain zones and routes in use by the reference architecture include: • zones for *.apps and *.sys (required) • a route for Ops Manager (required) • a route for Doppler (required) • a route for Loggregator (required) • a route for ssh access to app containers (optional) • a route for tcp access to tcp routers (optional)
Ops Manager	Deployed on the infrastructure network and accessible by FQDN or through an optional jumpbox.
BOSH Director	Deployed on the infrastructure network.
Application Load Balancer	Required. Load balancer that handles incoming HTTP, HTTPS, TCP, and SSL traffic and forwards them to the Gorouters. Load balancers are outside the scope of this document.
SSH Load Balancer	Optional. Load balancer that provides SSH access to application containers for developers. Load balancers are outside the scope of this document.
Gorouters	Accessed through the Application Load Balancer. Deployed on the Pivotal Application Service (PAS) network, one per AZ.
Diego Brains	This component is required. However, the SSH container access functionality is optional and enabled through the SSH Load Balancers. Deployed on the PAS network, one per AZ.
TCP Routers	Optional feature for TCP routing. Deployed on the PAS network, one per AZ.
CF Database	Reference architecture uses internal MySQL.
Storage Buckets	Reference architecture uses customer provided blobstore. Buckets are needed for BOSH and PAS.
Service Tiles	Deployed on the services network.
Service Accounts	Two service accounts are recommended: one for OpenStack "paving," and the other for Ops Manager and BOSH. Consult the following list: • Admin Account: Concourse will use this account to provision required OpenStack resources as well as a Keystone service account. • Keystone Service Account: This service account will be automatically provisioned with restricted access only to resources needed by PCF.
OpenStack Quota	The default compute quota on a new OpenStack subscription is typically not enough to host a multi-AZ deployment. The recommended quota for instances is 100. Your OpenStack network quotas may also need to be increased.

OpenStack Objects

The following table lists the network objects in this reference architecture.

Network Object	Notes	Estimated Number
Floating IP addresses	Two per deployment, one assigned to Ops Manager, the other to your jumpbox.	2
Project	One per deployment. A PCF deployment exists within a single project and a single OpenStack region, but should distribute PCF jobs and instances across three OpenStack AZs to ensure a high degree of availability.	1
Networks	The reference architecture requires the following Tenant Networks: • 1 x (/24) **Infrastructure** (Ops Manager, BOSH Director, Jumpbox). • 1 x (/20) **PAS** (Gorouters, Diego Cells, Cloud Controllers, etc.). • 1 x (/20) **Services** (RabbitMQ, MySQL, Spring Cloud Services, etc.) • 1 x (/24) **On-demand services** (Various.) An Internet-facing network is also required: • 1 x **Public** network. **Note**: In many cases, the public network is an "under the cloud" network that is shared across projects.	5

<table>
<tr><td>Routers</td><td>This reference architecture requires one router attached to all networks:
VirtualRouter: This router table enables the ingress/egress routes from/to Internet to the project networks and provides sNAT services.</td><td>1</td></tr>
<tr><td>Security Groups</td><td>The reference architecture requires one Security Groups. The following table describes the Security Group ingress rules:
<table>
<tr><th>Security Group</th><th>Port</th><th>From CIDR</th><th>Protocol</th><th>Description</th></tr>
<tr><td>OpsMgrSG</td><td>22</td><td>0.0.0.0/0</td><td>TCP</td><td>Ops Manager SSH access</td></tr>
<tr><td>OpsMgrSG</td><td>443</td><td>0.0.0.0/0</td><td>TCP</td><td>Ops Manager HTTP access</td></tr>
<tr><td>VmsSG</td><td>ALL</td><td>VPC_CIDR</td><td>ALL</td><td>Open up connections among BOSH-deployed VMs</td></tr>
</table>
Additional security groups may be needed which are specific to your chosen load balancing solution.</td><td>5</td></tr>
<tr><td>Load Balancers</td><td>PCF on OpenStack requires a load balancer, which can be configured with multiple listeners to forward HTTP/HTTPS/TCP traffic. Two load balancers are recommended: one to forward the traffic to the Gorouters, <code>AppsLB</code>, the other to forward the traffic to the Diego Brain SSH proxy, <code>SSHLB</code>.

The following table describes the required listeners for each load balancer:
<table>
<tr><th>Name</th><th>Instance/Port</th><th>LB Port</th><th>Protocol</th><th>Description</th></tr>
<tr><td>AppsLB</td><td>gorouter/80</td><td>80</td><td>HTTP</td><td>Forward traffic to Gorouters</td></tr>
<tr><td>AppsLB</td><td>gorouter/80</td><td>443</td><td>HTTPS</td><td>SSL termination and forward traffic to Gorouters</td></tr>
<tr><td>SSHLB</td><td>diego-brain/2222</td><td>2222</td><td>TCP</td><td>Forward traffic to Diego Brain for container SSH connections</td></tr>
</table>
Each load balancer needs a check to validate the health of the back-end instances:
<code>AppsLB</code> checks the health on Gorouter port 80 with TCP
<code>SSHLB</code> checks the health on Diego Brain port 2222 with TCP
Note: In many cases, the load balancers are provided as an “under the cloud” service that is shared across projects.</td><td>2</td></tr>
<tr><td>Jumpbox</td><td>Optional. Provides a way of accessing different network components. For example, you can configure it with your own permissions and then set it up to access to Pivotal Network to download tiles. Using a jumpbox is particularly useful in IaaSes where Ops Manager does not have a public IP address. In these cases, you can SSH into Ops Manager or any other component through the jumpbox.</td><td>1</td></tr>
</table>

vSphere Reference Architecture

Page last updated:

This guide presents reference architectures for Pivotal Cloud Foundry (PCF) on vSphere.

Overview

Pivotal validates the reference architectures described in this topic against multiple production-grade usage scenarios. These designs are sized for up to 1500 app instances.

This document does not replace the basic installation documentation, but gives proven examples of how to apply those instructions to real-world production environments.

PCF Products Validated	Version
PCF Ops Manager	2.0.latest
Pivotal Application Service (PAS)	2.0.latest

See PCF on vSphere Requirements for general requirements for running PCF and specific requirements for running PCF on vSphere.

PCF Reference Architectures

A PCF reference architecture describes a proven approach for deploying Pivotal Cloud Foundry on a specific IaaS, such as AWS, that meets the following requirements:

- Secure
- Publicly-accessible
- Includes common PCF-managed services such as MySQL, RabbitMQ, and Spring Cloud Services
- Can host at least 100 app instances, or far more

Pivotal provides reference architectures to help you determine the best configuration for your PCF deployment.

Base vSphere Reference Architecture

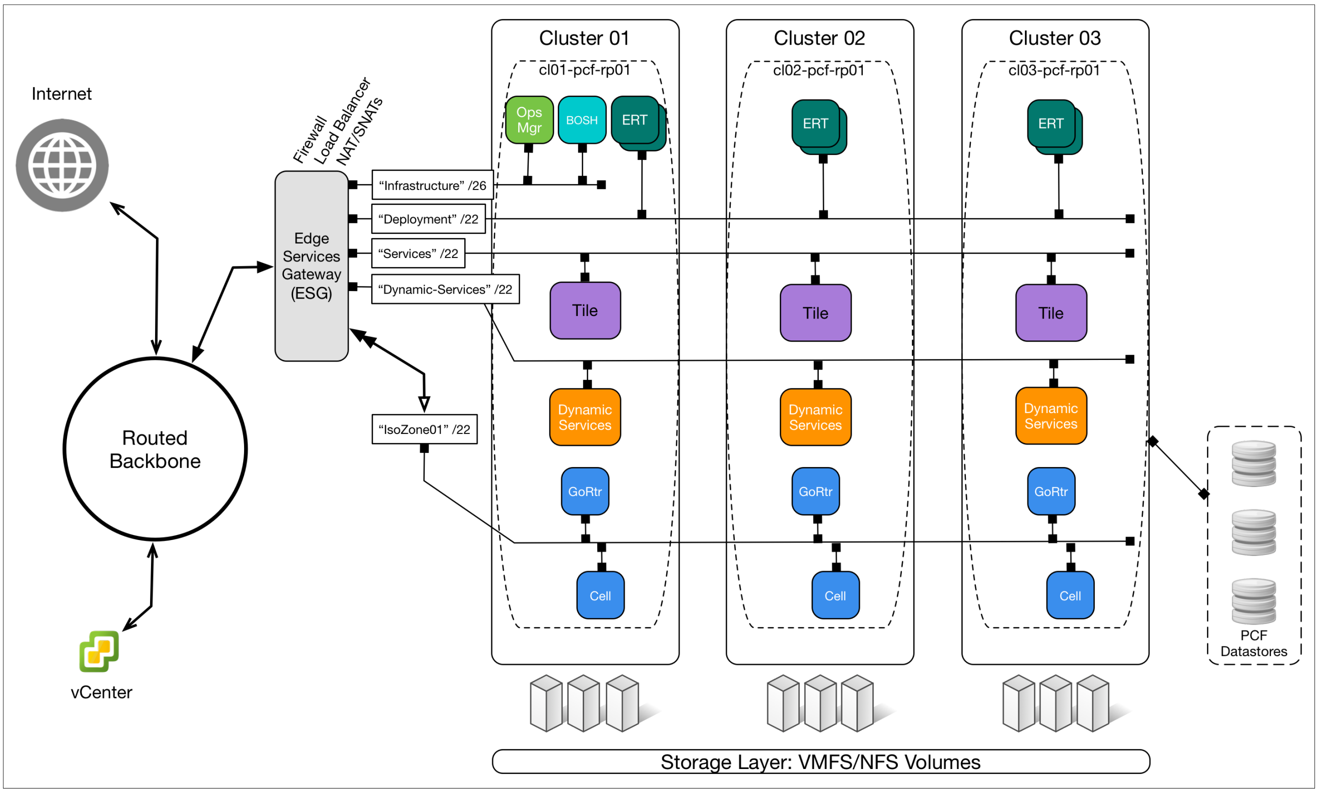
This recommended architecture includes VMware vSphere, NSX-V, and the ESG (Edge Services Gateway), a software-defined network services gateway that runs on VMware ESXi virtual hosts and combines routing, firewall, NAT/SNAT, and load balancing. In the absence of NSX, see below for architectures that do not rely on the ESG.

To use all features listed here, ESG requires at least **Advanced** licensing from VMware.

For more information about installing and configuring ESG for use with PCF on vSphere, see the Cookbook for Using Edge Services Gateway on PCF and VMware NSX.

The diagram below shows an architecture for one PCF installation in vSphere clusters, segmented with Resource Pools. More Resource Pools can be added to the existing Clusters to stack more PCF installations into the same capacity.

This design supports long-term use, capacity growth at the vSphere level, and maximum installation security through the ESG firewall. It allocates a minimum of three or more servers to each cluster, as recommended by vSphere, and spreads PCF components across three or another odd number of clusters, as recommended for High Availability.



View a larger version of this diagram.

Installation

To create a system following this architecture, perform the following steps:

1. In vCenter, create or identify three existing clusters.
2. Enable DRS on each cluster and set vMotion to **fully automated**. Populate each cluster with a Resource Pool for each PCF installation.
3. For compute, populate each cluster with three or more ESXi hosts, making nine or more hosts for each installation. All installations collectively draw from the same group of nine or more hosts.
4. In the PCF deployment, use Ops Manager to create three Availability Zones (AZs), each corresponding to one of the Resource Pools from each cluster.
5. For storage, add dedicated datastores to each PCF deployment following one of the two approaches, vertical or horizontal, as described in the Storage Configuration section of this topic.
6. Supply core networking for each deployment by configuring an ESG with the following subnets. See the Networking Design section of this topic for details.
   - Infrastructure
   - PAS
   - Service tiles (one or more)
   - Dynamic service tiles (a network managed entirely by BOSH Director)
   - IsoZone##
   - Container-to-Container Networking

   Pivotal recommends NSX Logical Switches (vWires) for all networks used by PCF. This approach avoids VLAN consumption while benefiting from the overlay capability NSX enables. NSX can create a DPG (Distributed Port Group) on a DVS (Distributed Virtual Switch) for each interface provisioned on the ESG as shown in the Port Groups diagram below.

   Alternatively, port groups on a DVS with VLANs tagged on each can be used for the networks above.

Scaling

You can scale up this architecture to support additional PCF installations with the same capacity, keeping each resource-protected and separated.

To support more PCF installations, scale this architecture vertically by adding Resource Pools to existing clusters. To add capacity to all PCF installations, scale it horizontally by adding ESXi hosts to the existing clusters in sets of three, one per cluster.

Priority

In this architecture, multiple PCF installations can share host resources. You can use vCenter resource allocation shares to assign `High`, `Normal`, or `Low` priority to pools used by each installation. When host resources keep up with demand, these share values make no difference, but when multiple installations compete for limited resources, you can prioritize a production installation over a development installation by assigning its resource pools a `High` share value setting.

Storage Configuration

Shared storage is a requirement for PCF. You can allocate networked storage to the host clusters following one of two common approaches, *horizontal* or *vertical*. The approach you follow should reflect how your data center arranges its storage and host blocks in its physical layout.

- **Horizontal**: You grant all hosts access to all datastores and assign a subset to each installation.

  For example, with six datastores `ds01` through `ds06`, you grant all nine hosts access to all six datastores. You then provision your first PCF installation to use stores `ds01` through `ds03`, and your second PCF installation to use `ds04` through `ds06`.
- **Vertical**: You grant each cluster its own dedicated datastores, creating a "cluster-aligned" storage strategy. vSphere VSAN is an example of this architecture.

  For example, with six datastores `ds01` through `ds06`, you assign datastores `ds01` and `ds02` to your first cluster, `ds03` and `ds04` to your second cluster, and `ds05` and `ds06` to your third cluster. You then provision your first PCF installation to use `ds01`, `ds03`, and `ds05`, and your second PCF installation to use `ds02`, `ds04`, and `ds06`. With this arrangement, all VMs in the same installation and cluster share a dedicated datastore.

Note: If a datastore is part of a vSphere Storage Cluster using sDRS (storage DRS), you must disable the s-vMotion feature on any datastores used by PCF. Otherwise, s-vMotion activity can rename independent disks and cause BOSH to malfunction. For more information, see How to Migrate PCF to a New Datastore in vSphere.

Storage Capacity and Type

Capacity

Pivotal recommends the following capacity allocation for PCF installations:

- For production use, at least 8 TB of data storage, either as one 8 TB store or a number of smaller volumes adding up to 8 TB. Frequently-used development may require significantly more storage to accomodate new code and buildpacks.
- For small installations without many tiles, 4-6 TB.

The primary consumer of storage is the NFS/WebDAV blobstore.

Note: PCF does not currently support using vSphere Storage Clusters with the latest versions of PCF validated for the reference architecture. Datastores should be listed in the vSphere tile by their native name, not the cluster name created by vCenter for the storage cluster.

Type

Pivotal recommends the following: - Either block-based (fiber channel or iSCSI) and file-based (NFS) over high-speed carriers such as 8Gb FC or 10GigE - Redundant storage for both ephemeral and persistent storage types used by PCF. All-flash-based storage and SSD-cached storage are highly desirable. - Data de-duplication and compression, if done in hardware at the storage array

These technologies can dramatically reduce the need for storage capacity.

Networking

Using VMware NSX SDN (software-defined networking) provides the following benefits:

- Network fencing of the entire PCF installation
- Distributed, hyper-local firewall capability per installation through the built-in ESG Firewall
- High capacity, resilient, distributed load balancing per installation through the ESG Load Balancer
- Element obfuscation through the use of non-routed RFC-1918 networks behind the ESG and the use of SNAT/DNAT connections to expose only the endpoints of Cloud Foundry that require exposure
- High-repeatability of installations through the reuse of all network and addressing conventions on the right-hand side of the diagram, the **Tenant Side**
- Centrally-managed rule and ACL sharing through the NSX Manager Global Ruleset
- HA pairs of ESGs (optional) for extra levels of redundancy
- BOSH CPI can add and remove Gorouter members from load balanced pools in ESG (not an Ops Manager feature)
- ESG Security Group tagging managed by BOSH, to group like VMs per PCF installation into security groups

Warning: The overlay network IP range for Container-to-Container Networking must not conflict with any other IP addresses in your network.

Note: When using VMware NSX for vSphere 6.2.3+, the default VXLAN port of 4789 used by Container-to-Container Networking will not work. To resolve this issue, override the default by navigating to the Networking section of the PAS tile and entering a different value in **VXLAN Tunnel Endpoint Port**.

Networking Design

Each PCF installation consumes four or more networks with the ESG, aligned to the following specific job types:

- **Infrastructure**: This small network hosts resources that interact with the IaaS layer and back-office systems, such as the cloud provider interface (CPI), BOSH, Ops Manager, and other utility VMs such as jumpbox VM. Operators access these resources to manage a PCF installation.
- **Deployment**: Also known as the *apps wire*, this network has a large CIDR range. It hosts the PAS tile, Diego Cells, and Windows Cells, and is the network onto which apps are deployed.
- **Services**: This network has a large CIDR range. It hosts service tiles that are installed using Ops Manager and managed by BOSH. One approach is to use this network for all PCF tiles except PAS.
- **Dynamic Services**: A single network granted to BOSH Director for use with service tiles that require an on-demand, or *dynamic*, address space for deployment. This is a special purpose network that is marked as “Services” with a checkbox in the vSphere Ops Manager Director tile. See About the On-Demand Services SDK for more information.
- **IsoZones##**: A single network granted to the PCF Isolation Segment tile for use to isolate Gorouters and Diego Cells into a network space independent of the PAS installation.
- **Container-to-Container Networking**: A single network dedicated to communication between containers.

Note: All of these networks are considered *inside* or *tenant-side* networks, and use non-routable RFC-1918 network space which are not advertised to the outside by the ESG to make provisioning repeatable. The ESG routes between the tenant and service provider side networks and connects traffic through using SNAT and DNAT.

For each PCF installation, you should provision an ESG with at least four routable IP addresses from the service provider:

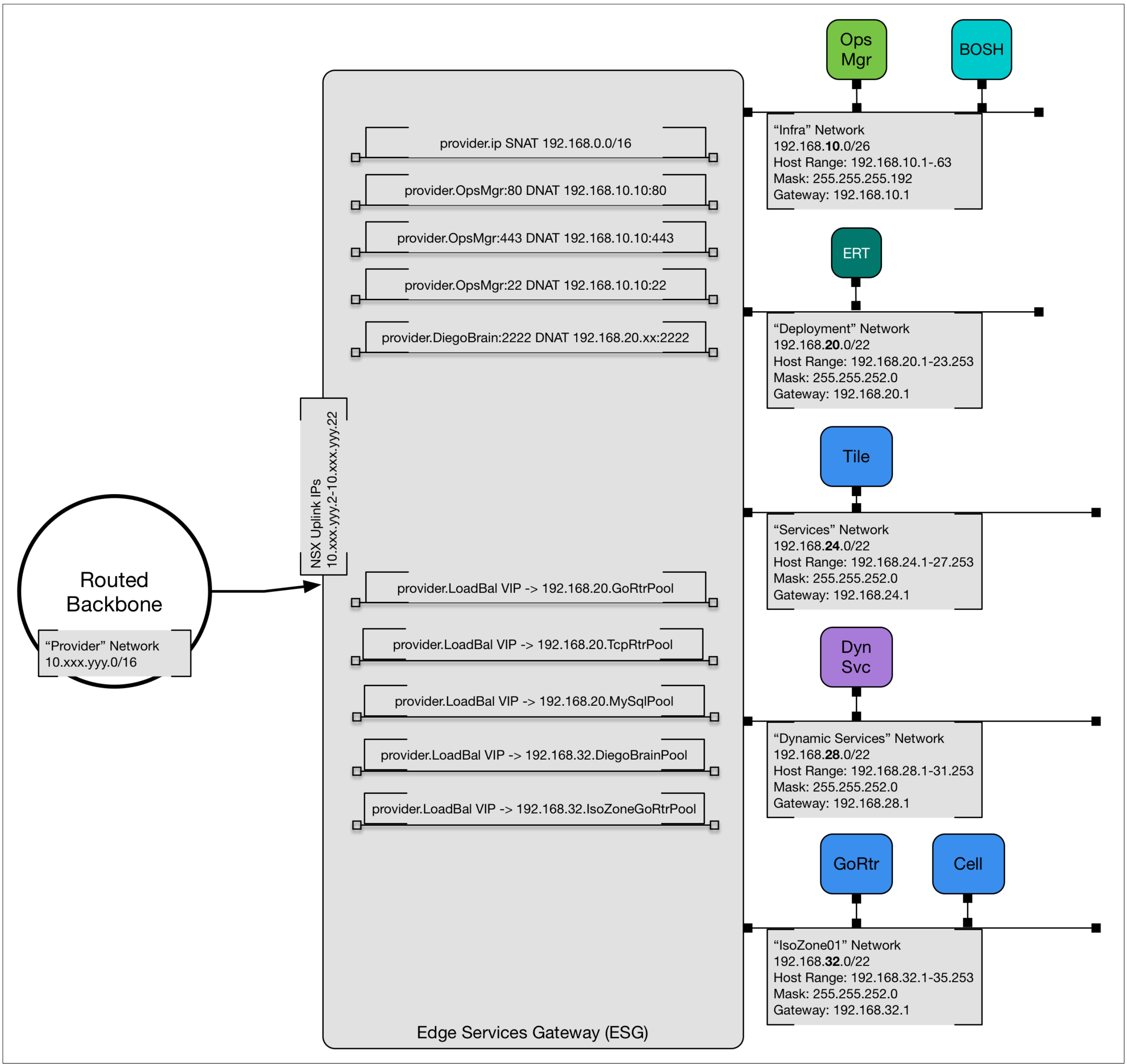
- A static IP address by which NSX Manager manages the ESG
- A static IP address for use as egress SNAT. Traffic from the tenant side exits the Edge on this IP address
- A static IP address for DNATs to Ops Manager
- A static IP address for the load balancer VIP that balances to a pool of PCF Gorouters (HTTP/HTTPS)

In addition to these, many more uses for IP addresses on the routed side of the ESG exist. Pivotal recommends reserving a total of ten contiguous static IP addresses per ESG for future needs and flexibility. Examples include the following:

- Load balancer VIP for TCP Routers, if deploying TCP routing for non 80/443 access to apps
- Load Balancer VIP for DiegoBrains, if deploying in multiples
- Load Balancer VIP for MySQL proxies, if deploying in multiples
- Monitoring or metrics endpoint for platform monitoring

On the tenant side, each interface defined on the ESG acts as the IP gateway for the network used. Pivotal recommends allocating the following address ranges for the networks, and defining the gateway at `192.168.zzz.1` for each:

- **Infrastructure** network: 192.168.10.0/26
- **Deployment** network: 192.168.20.0/22
- **Services** network: 192.168.24.0/22
- **Dynamic Services** network: 192.168.28.0/22
- **IsoZone##** network: 192.168.32.0/22
- **Container-to-Container Networking** network: 10.255.0.0/16



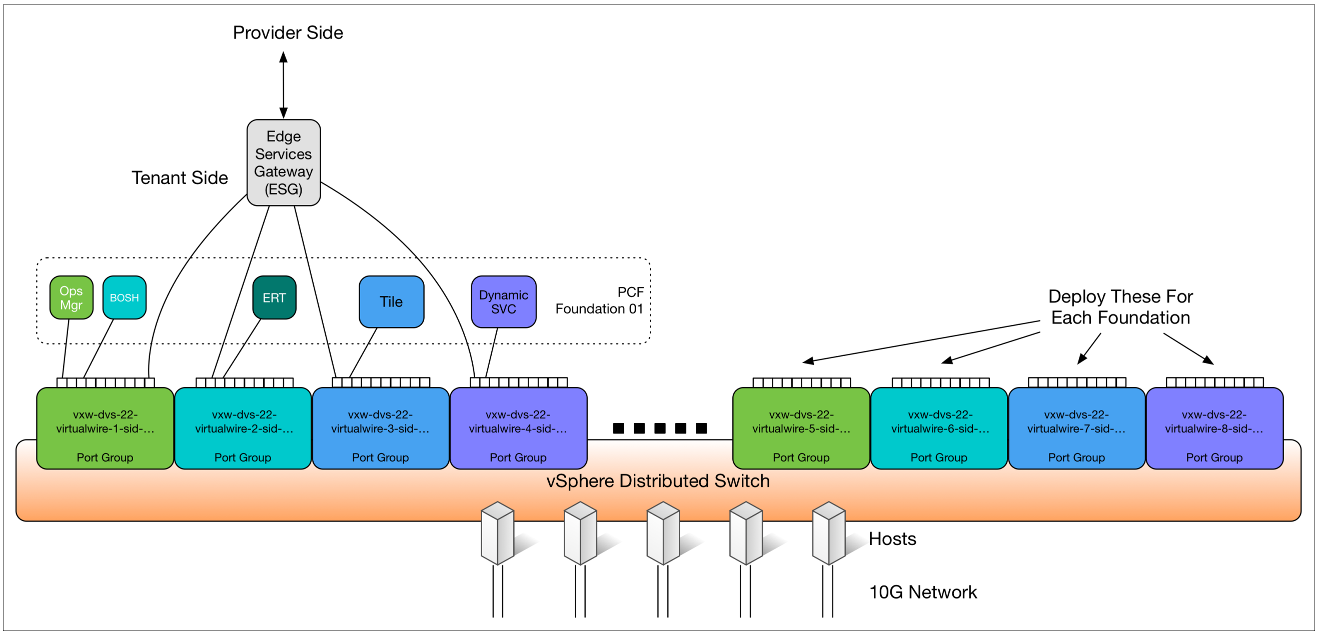
View a larger version of this diagram.

Note: Review the container-to-container settings in your installation for any IP address range conflicts with the enterprise addressing scheme in use by your organization. The specified range cannot be used anywhere else in your deployment. Change the range to a non-conflicting range as needed.

Distributed Port Groups

Pivotal recommends vSphere DVS for all Clusters used by PCF. NSX creates a DPG for each interface provisioned on the ESG.

NSX Logical Switches should be used on the Tenant Side of this design, which leverages vWires, reducing the dependency on available VLAN capacity.



View a larger version of this diagram.

High Performance Variants

One-Armed Load Balancing

The ESG can act as a stand-alone, one-armed load balancer.

This variant can improve performance and separate the dependence on the ESG that acts as NAT/SNAT/Firewall/Gorouter by separating the load balancing function to a separate ESG deployed exclusively for use per installation.

This variant divides the jobs between two ESGs per install rather than one. To implement this architecture, you place a single interface (internal) of a new ESG on the Deployment network, enable the load balancing function, and DNAT to it through the boundary ESG.

Reference Architecture Without VMware NSX

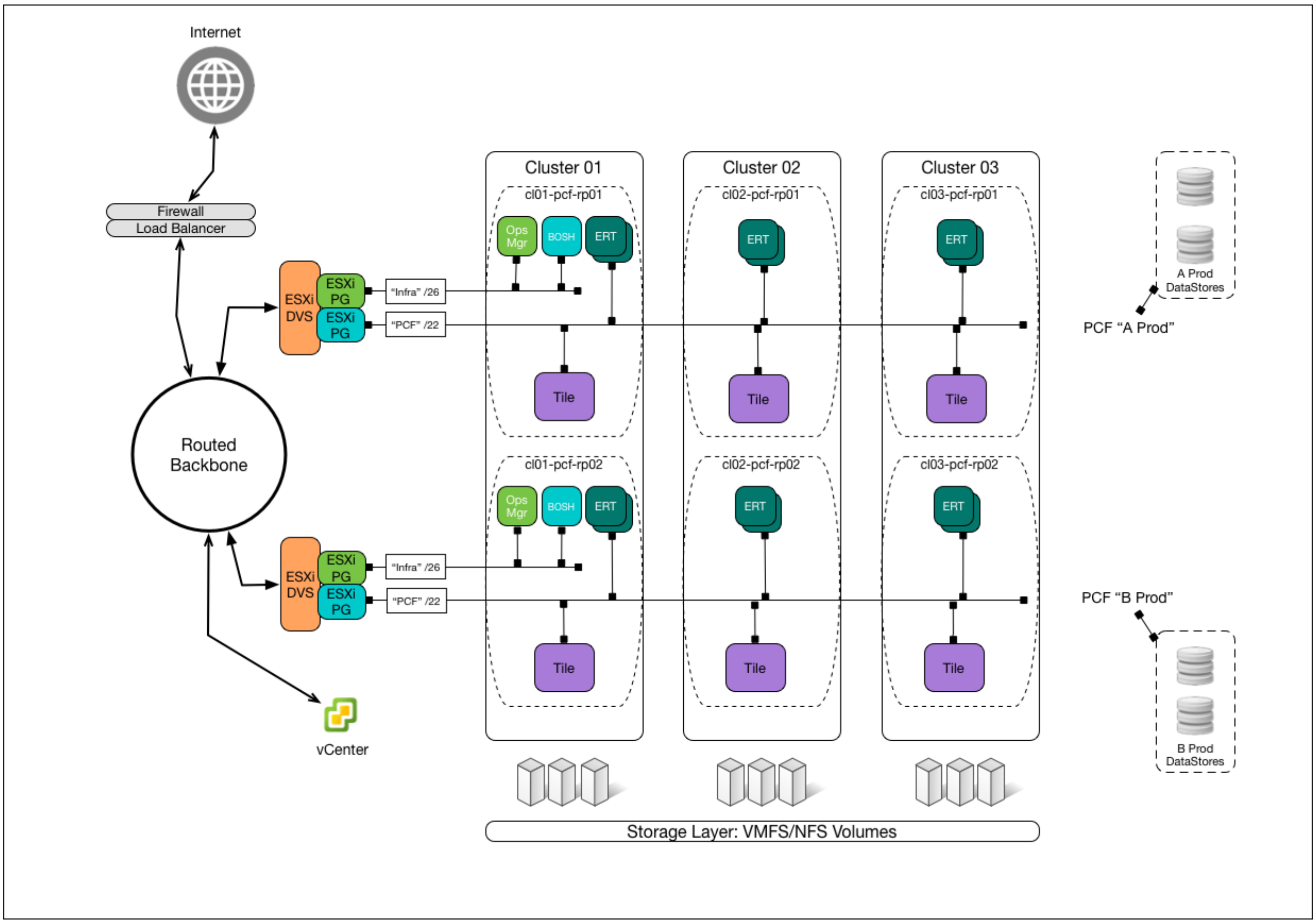
The reference architecture for deploying production PCF on vSphere without VMware NSX SDN technology follows the base architecture, but with the following differences.

Networking Features

- Load balancing is handled by an external service, such as a hardware appliance or a VM from a third party.
- An external service also performs SSL termination unless that is passed through to the Gorouters.
- You must set up firewalls for each zone or network inside the installation, rather than having the ESG fence all the inside networks.
- There is no network fencing of the PCF installation, so RFC-1918 non-routable networks are not used, DNAT/SNAT is not used, and the address space consumed is from routable ranges already established in the datacenter.

Networking Design

The more traditional approach without SDN is to deploy a single VLAN for use with all of a PCF installation, or a pair of VLANs, one for infrastructure and one for the rest of the PCF installation. As VLAN capacity can be limited and scarce, this design seeks to limit the need for VLANs to a functional minimum.



View a larger version of this diagram.

In this example, the firewall and load balancer functions run outside of vSphere, on generic devices that most datacenters provide. The PCF installation is bound to two port groups provided by a DVS on ESXi, each of which aligns to different job types:

1. **Infra**: CPI, BOSH, and Ops Manager VMs that communicate with the IaaS layer

2. **PCF**: The deployment network for all tiles, including PAS

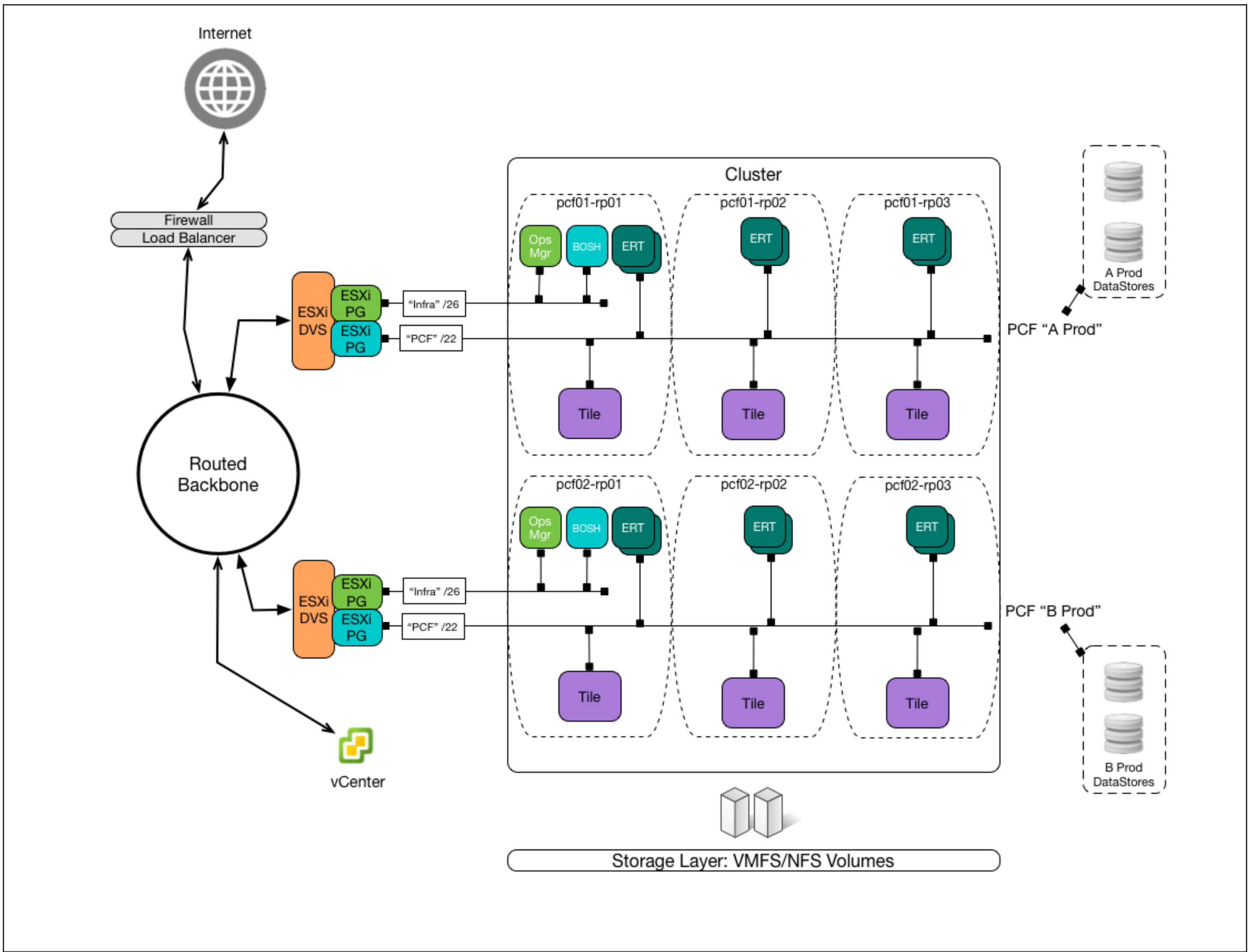
In a typical installation, you assign each of these port groups to a VLAN from the datacenter pool, and a routable IP address segment. Routing functions are handled by switching layers outside of vSphere, such as a top-of-rack or end-of-row switch/router appliance.

It is valid to deploy more networks than these two, up to and including those shown in the original design, so deploy them if the resources are readily available. The main thing to keep in mind is that this is a requirement per PCF installation, so keep a count of how many of those overall you will require.

Reference Architecture Without Multiple Clusters

If you are working with three or more ESXi hosts and want to use less resources than the base architecture requires, Pivotal recommends setting up PCF in three clusters with one host in each. The key point is to try to get to three AZs if possible, as this reduces future difficulties when increasing the size of your PCF installation.

To reduce resource use even further, you can place all hosts into a single cluster with VMware DRS and high availability enabled. The following is an example of that approach, using resource pools in the single cluster to emulate a three-cluster design. While the resource pools do little more that organize the VMs, it forces a habit of deploying PCF constructs in threes as recommended by Pivotal.



View a larger version of this diagram.

You may be tempted to consider a two-cluster architecture. A two-cluster architecture may offer useful symmetry at the vSphere level, but PCF works best when it deploys resources in odd numbers. A two-cluster configuration forces the operator into aligning odd-numbered components into even numbered containers, which does not work well for PCF internal voting algorithms. If you do not want to consume three clusters for PCF, using one cluster works better than using two.

Networking Design

For a single-cluster deployment, follow the networking setup described in either the base or the without-NSX. architectures above. The internal compute arrangement for a production PCF deployment does not affect its networking.

Pivotal recommends mapping all datastores used by PCF to all of the hosts in a single-cluster deployment.

Multi-Datacenter Reference Architecture

To avoid downtime, some PCF customer scenarios demand a multi-datacenter architecture that spreads deployment resources across more than one physical location. A multi-datacenter architecture can support the hardware, power source, and geographic redundancy needed to guarantee high availability.

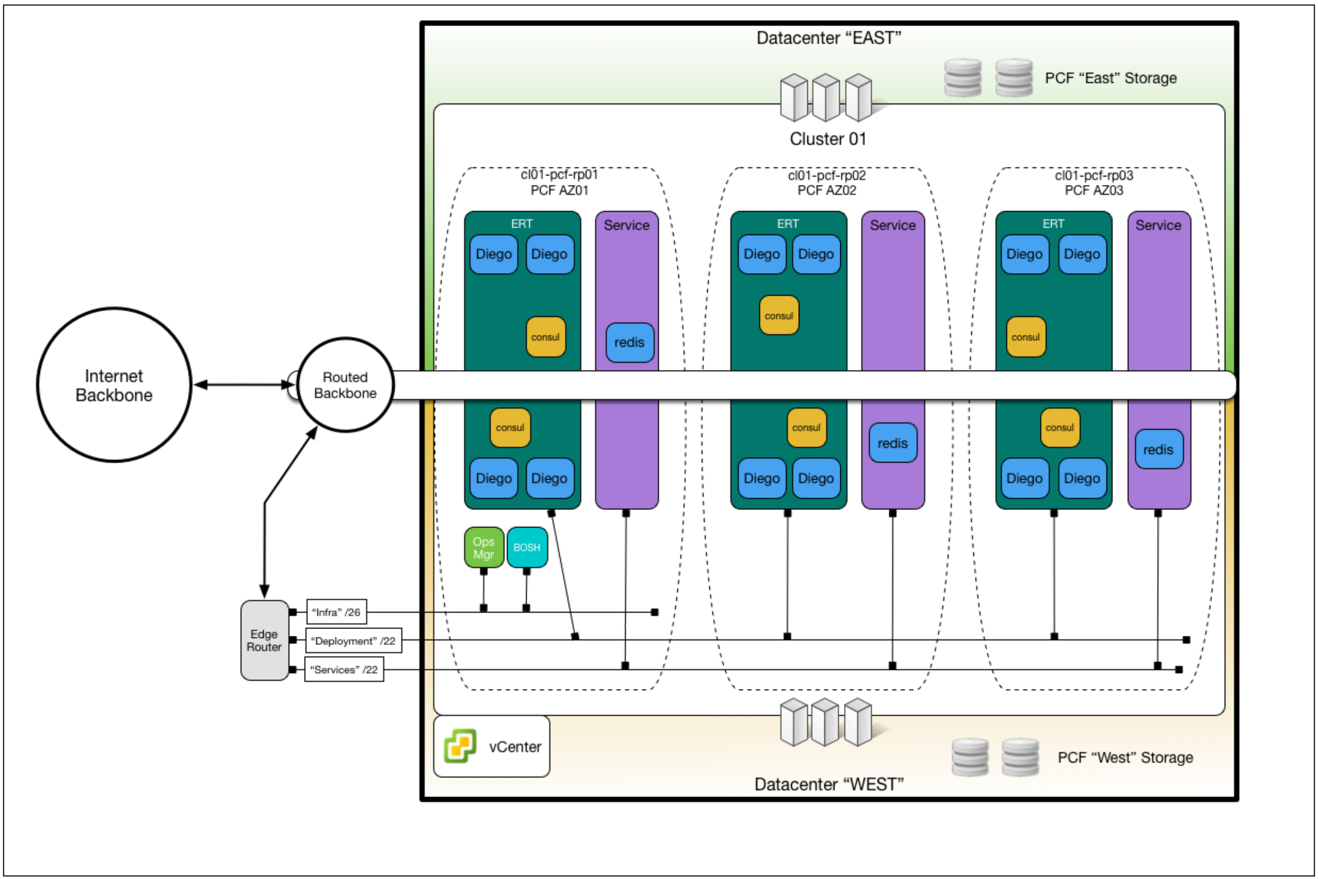
One strategy for high availability is to keep a record of how many hosts are in a cluster and deploy enough copies of a PCF component in that AZ to ensure survivability in a site loss. This means placing large, odd numbers of components in the cluster so that at least two components are left on either site in the event of a site outage. In a four host cluster, this calls for five VMs, so each site has at least two. DRS anti-affinity rules can be used here, set at the IaaS level, to force like VMs apart for best effect.

The two main ways of designing a multi-datacenter PCF architecture is with stretched clusters, in which single logical clusters combine components in multiple physical locations, and East/West clusters, in which locally self-contained clusters are mirrored across multiple locations.

Both of these approaches have their own caveats, and you can combine either with the without-NSX and single-cluster architectures described above.

Multi-Datacenter vSphere With Stretched Clusters

For this approach, you define logical clusters that contain components physically located in two or more sites. With four hosts, for example, build a a four-host cluster with two hosts in an East datacenter and two from the West. Apply networking such that all hosts see the same networks through a stretched layer 2 application, or use NSX or another SDN solution to extend L2 networking via L3 connections.



View a larger version of this diagram.

PCF and BOSH treat the stretched cluster as an AZ, and make the same demands on it that they do with any other AZ. The hosting, networking, and storage components within the stretched cluster must perform with normal latency and connectivity.

For seamless operation, hosts must share all datastores, and you need to replicate storage across sites. Otherwise, vMotion cannot move VMs freely across hosts for maintenance or DRS.

A stretched version of the base architecture splits three clusters across two sites, yielding a 4×3×3 geometry:

- Four hosts per cluster (two from each site)
- Three clusters for PCF as AZs
- Three AZs mapped to PCF clusters

You can also deploy a stretched version of the single-cluster model. This may be the more practical approach to achieving HA, since any stretched deployment already requires so many resources from two sites.

As with any VMware installation, job scheduling works more efficiently when VMs have fewer cores, so you should configure many smaller Diego Cell VMs rather than a lower number of larger ones. If 2-core to 4-core VMs can handle your apps, favor them over 8- and 12-core options. This is especially important with stretched deployments.

Network traffic is a challenge with stretched clusters, since app traffic may enter at any connection point in either location, but can only leave through a designated gateway. The architect should consider that app traffic landing in the East might have to flow out of the West, a "trombone effect" that forces additional traffic across datacenter links.

Multi-Datacenter vSphere With Combined East/West Clusters

For this approach, the architect assigns parallel capacity from two sites independently, and deploys clusters to PCF in matched pairs. This creates even numbers of clusters, which makes suboptimal use of resources in PCF.

East/West mirroring the base architecture yields a deployment with six total clusters, three from each side. This may seem like a lot of gear to apply to PCF, but in a Business Continuity and Disaster Recovery (BCDR) scenario, doubling everything is the point.

Combining the East/West multi-datacenter and single-cluster approaches creates a geometry with two clusters and pools in one cluster per site, or six AZs. Such a deployment only uses one cluster of capacity from each site, and does not scale readily. But drawing capacity from only one cluster makes it easy to provision with only a few hosts.

A multi-datacenter architecture makes replicating storage less critical. There are enough AZs from either side to survive a point failure, and you can recover the installation without vSphere HA enabled for the clusters.

Additional Documentation

- How to Upgrade vSphere without PCF Downtime
- How to Migrate PCF to a New Datastore in vSphere

Using Edge Services Gateway on VMware NSX

Page last updated:

This cookbook provides guidance on how to configure the NSX firewall, load balancing and NAT/SNAT services for Pivotal Cloud Foundry (PCF) on vSphere installations. These NSX-provided services take the place of an external device or the bundled HAProxy VM in PCF.

This document presents the reader with fundamental configuration options of an Edge Services Gateway (ESG) with PCF and vSphere NSX. Its purpose is not to dictate the settings required on every deployment, but instead to empower the NSX Administrator with the ability to establish a known good “base” configuration and apply specific security configurations as required.

If you are using NSX, the specific configurations described here supersede any general recommendations in the [Preparing Your Firewall](../../customizing/config_firewall.html) topic.

Assumptions

This document assumes that the reader has the level of skill required to install and configure the following products:

- VMware vSphere 5.5 or greater
- NSX 6.1.x or greater
- PCF 1.6 or greater

For detailed installation and configuration information about these products, refer to the following documents:

- vSphere Documentation
- NSX Installation and Upgrade Guide
- Reference Design: VMware NSX for vSphere (NSX) Network Virtualization Design Guide
- Pivotal Cloud Foundry Documentation

General Overview

This cookbook follows a three-step recipe to deploy PCF behind an ESG:

1. Configure Firewall
2. Configure Load Balancer
3. Configure NAT/SNAT

The ESG can scale to accommodate very large PCF deployments as needed.

This cookbook focuses on a single-site deployment and makes the following design assumptions:

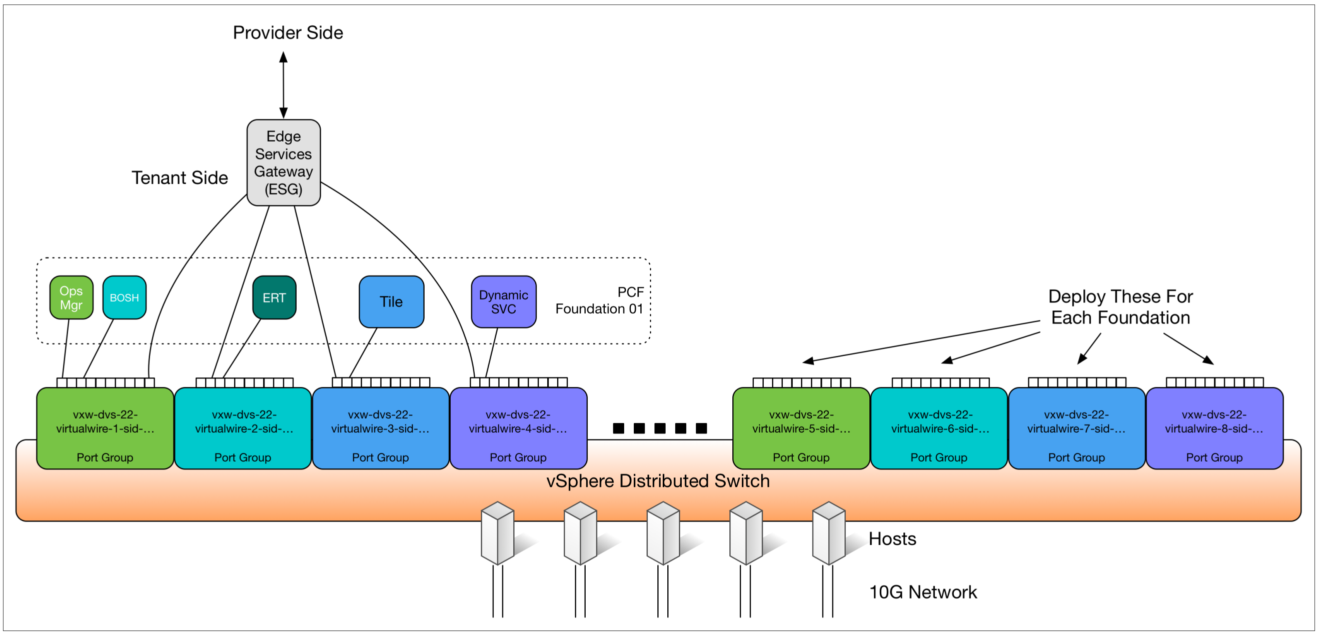
- There are five non-routable networks on the tenant (inside) side of the ESG.
  - The **Infra** network is used to deploy Ops Manager and BOSH Director.
  - The **Deployment** network is used exclusively by Pivotal Application Service (PAS) to deploy DEAs/Cells that host apps and related elements.
  - The **CF Tiles** network is used for all other deployed tiles in a PCF installation.
  - The **Services** network is used by BOSH Director for service tiles.
  - The **Container-to-Container** network is used for container to container communication in the Cells.
- There is a single service provider (outside) interface on the ESG that provides Firewall, Load Balancing and NAT/SNAT services.
- The service provider (outside) interface is connected appropriately to the network backbone of the environment, as either routed or non-routed depending on the design. This cookbook does not cover provisioning of the uplink interface.
- Routable IP addresses should be applied to the service provider (outside) interface of the ESG. Pivotal recommends that you apply 10 consecutive routable IP addresses to each ESG.
  - One reserved for NSX use (Controller to Edge I/F)
  - One for NSX Load Balancer to Gorouters
  - One for NSX Load Balancer to Diego Brains for SSH to apps
  - One routable IP address, used to access the Ops Manager frontend

- One routable IP address, used with SNAT egress
- Five for future use

Pivotal recommends that operators deploy the ESGs as high availability (HA) pairs in vSphere. Also, Pivotal recommends that they be sized “large” or greater for any pre-production or production use. The deployed size of the ESG impacts its overall performance, including how many SSL tunnels it can terminate.

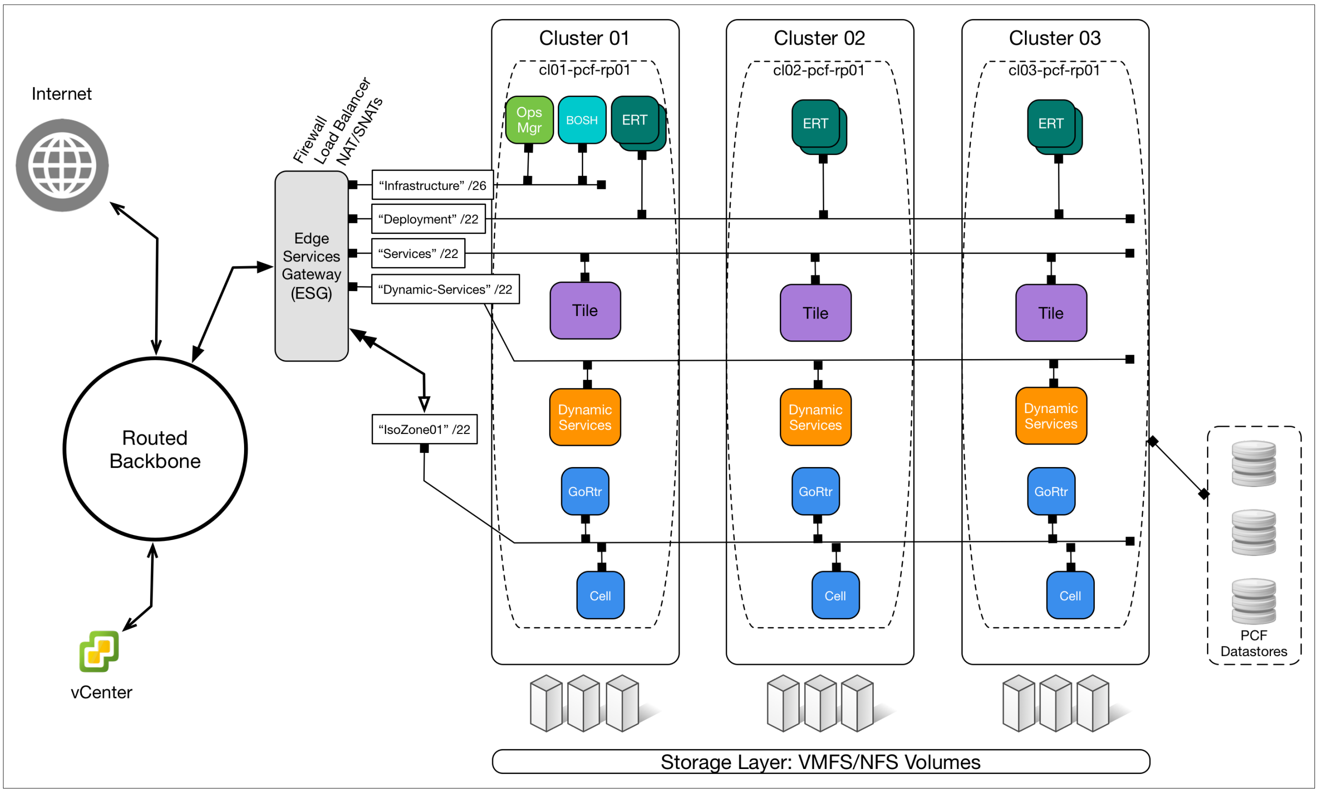
The ESGs have an interface in each port group used by PCF as well as a port group on the service provider (outside), often called the “transit network.” Each PCF installation has a set of port groups in a vSphere DVS to support connectivity, so that the ESG arrangement is repeated for every PCF install. It is not necessary to build a DVS for each ESG/PCF install. You do not re-use an ESG amongst PCF deployments. NSX Logical Switches (VXLAN vWires) are ideal candidates for use with this architecture.

The following diagram provides an example of port groups used with an ESG:



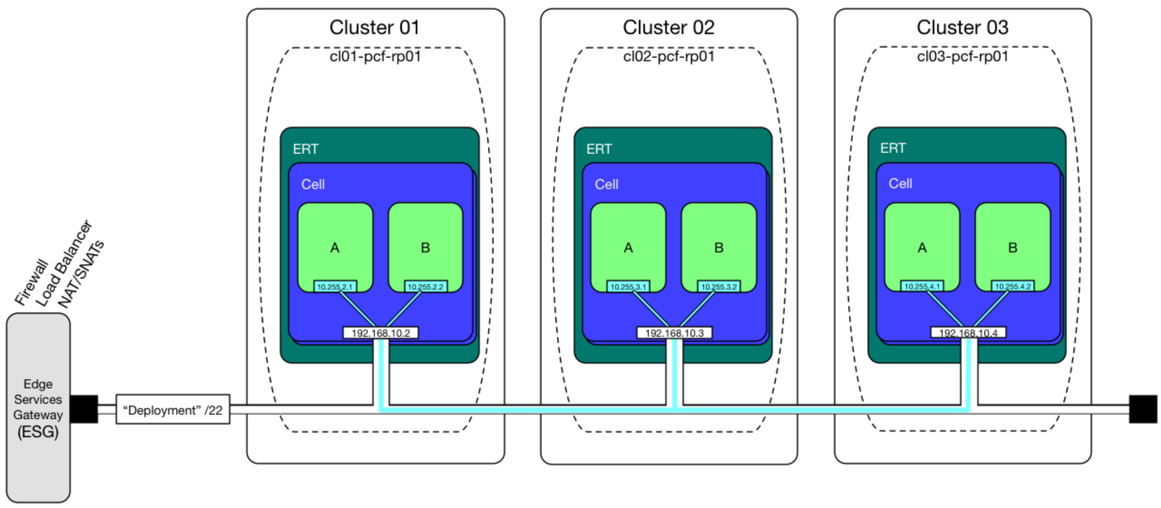
View a larger version of this diagram.

The following is an example of a network architecture deployment.



[View a larger version of this diagram](#).

The following diagram illustrates container-to-container networking. The overlay addresses are wrapped and transported using the underlay deployment subnet.



[View a larger version of this diagram](#).

Prep Step: Configure DNS and Network Prerequisites

As a prerequisite, create wildcard DNS entries for system and apps domains in PCF. Map these domains to the selected IP address on the uplink (outside) interface of the ESG in your DNS server.

The wildcard DNS `A` record must resolve to an IP address associated with the outside interface of the ESG for it to function as a load balancer. You can either use a single IP address to resolve both the system and apps domain, or one IP address for each.

In addition, assign the following IP addresses and address ranges within your network:

1. Assign IP Addresses to the "Uplink" (outside) interface
   - Typically you have one SNAT and three DNATs per ESG.
   - IP associated for SNAT use: All PCF internal IP addresses appear to be coming from this IP address at the ESG.
   - IP associated with Ops Manager DNAT: This IP address is the publicly routable interface for Ops Manager UI and SSH access.

2. Assign "Internal" Interface IP Address Space to the Edge Gateway.
   - 192.168.10.0/26 = PCF Deployment Network (Logical Switch or Port Group)
   - 192.168.20.0/22 = Deployment Network for PAS tile
   - 192.168.24.0/22 = CF Tiles Network for all Tiles besides PAS
   - 192.168.28.0/22 = Dynamic Services network for BOSH Director-managed service tiles.
   - 10.255.0.0/16 = Container-to-Container network for intercontainer communication.

Step 1: Configure Firewall

This procedure populates the ESG internal firewall with rules to protect a PCF installation.

These rules provide granular control on what can be accessed within a PCF installation. For example, rules can be used to allow or deny another PCF installation behind a different ESG access to apps published within the installation you are protecting.

This step is not required for the installation to function properly when the firewall feature is disabled or set to "Allow All."

To configure the ESG firewall, navigate to **Edge**, **Manage**, **Firewall** and set the following:

Name	Source	Destination	Service	Action
Allow Ingress -> Ops Manager	any	IP_of_OpsMgr	SSH, HTTP, HTTPS	Accept
Allow Ingress -> PAS	any	IP_of_NSX-LB	HTTP, HTTPS	Accept
Allow Ingress -> SSH for Apps	any	tcp:IP_of_DiegoBrain:2222	any	Accept
Allow Ingress -> TCProuter	any	tcp:IP_of_NSX-TCP-LB:5000	any	Accept
Allow Inside <-> Inside	192.168.10.0/26 192.168.20.0/22 192.168.24.0/22 192.168.28.0/22	192.168.10.0/26 192.168.20.0/22 192.168.24.0/22 192.168.28.0/22	any	Accept
Allow Egress -> IaaS	192.168.10.0/26	IP_of_vCenter IPs_of_ESXi-Svrs	HTTP, HTTPS	Accept
Allow Egress -> DNS	192.168.0.0/16	IPs_of_DNS	DNS, DNS-UDP	Accept
Allow Egress -> NTP	192.168.0.0/16	IPs_of_NTP	NTP	Accept
Allow Egress -> SYSLOG	192.168.0.0/16	IPs_of_Syslog:514	SYSLOG	Accept
Allow ICMP	192.168.10.0/26	*	ICMP	Accept
Allow Egress -> LDAP	192.168.10.0/26 192.168.20.0/22	IPs_of_LDAP:389	LDAP, LDAP-over-ssl	Accept
Allow Egress -> All Outbound	192.168.0.0/16	any	any	Accept
Default Rule	any	any	any	Deny

Step 2: Configure Load Balancer

The ESG provides software load balancing functionality, equivalent to the bundled HAProxy that is included with PCF, or hardware appliances such as an F5 or A10 load balancer.

This step is required for the installation to function properly.

There are seven high level steps to this procedure:

1. Import SSL certificates to the Edge for SSL termination.

2. Enable the load balancer.
3. Create Application Profiles in the Load Balancing tab of NSX.
4. Create Application Rules in the Load Balancer.
5. Create Service Monitors for each pool type.
6. Create Application Pools for the multiple groups needing load balancing.
7. Create a virtual server (also known as a VIP) to pool balanced IP addresses.

What you will need:

- PEM files of SSL certificates provided by the certificate supplier for only this installation of PCF, or the self-signed SSL certificates generated during PCF installation.

In this procedure you marry the ESG's IP address used for load balancing with a series of internal IP addresses provisioned for Gorouters in PCF. It is important to know the IP addresses used for the Gorouters beforehand.

These IP addresses can be pre-selected or reserved prior to deployment (recommended) or discovered after deployment by looking them up in BOSH Director, which lists them in the release information of the PAS installation.

Step 2.1: Import SSL Certificate

PCF requires SSL termination at the load balancer.

> **Note**: If you intend to pass SSL termination through the load balancer directly to the Gorouters, you can skip the step below and select **Enable SSL Passthru** on the **HTTPS Application Profile**.

To enable SSL termination at the load balancer in ESG, access the ESG UI and perform the following steps:

1. Select **Edge**, **Manage**, **Settings**, and then **Certificates**.
2. Click Green Plus button to Add Certificate.
3. Insert PEM file contents from the Networking configuration screen of PAS.
4. Save the results.

Step 2.2: Enable the Load Balancer

To enable the load balancer, access the ESG UI and perform the following steps:

1. Select **Edge**, **Manage**, **Load Balancer**, and then **Global Configuration**.
2. Edit load balancer global configuration.
3. Enable load balancer.
4. Enable acceleration.
5. Set logging to desired level (**Info** or greater).

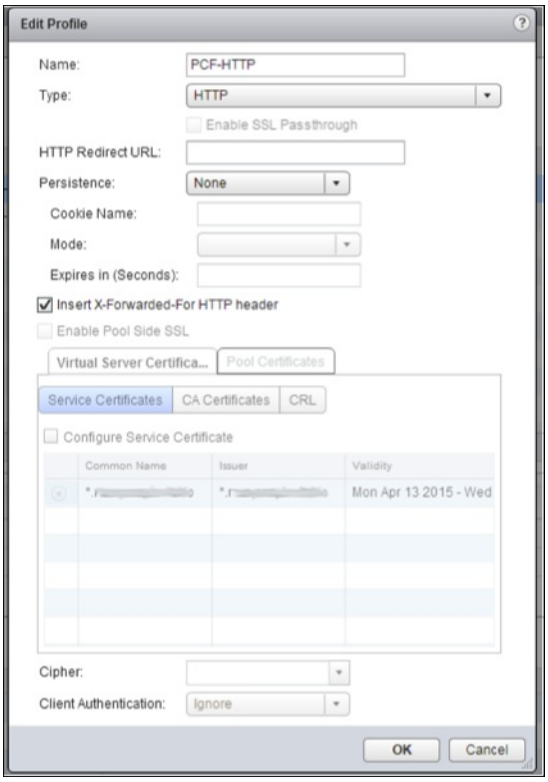
Step 2.3: Create Application Profiles

The Application Profiles allow advanced `x-forwarded` options as well as linking to the SSL Certificate. You must create three Profiles: **PCF-HTTP**, **PCF-HTTPS** and **PCF-TCP**.

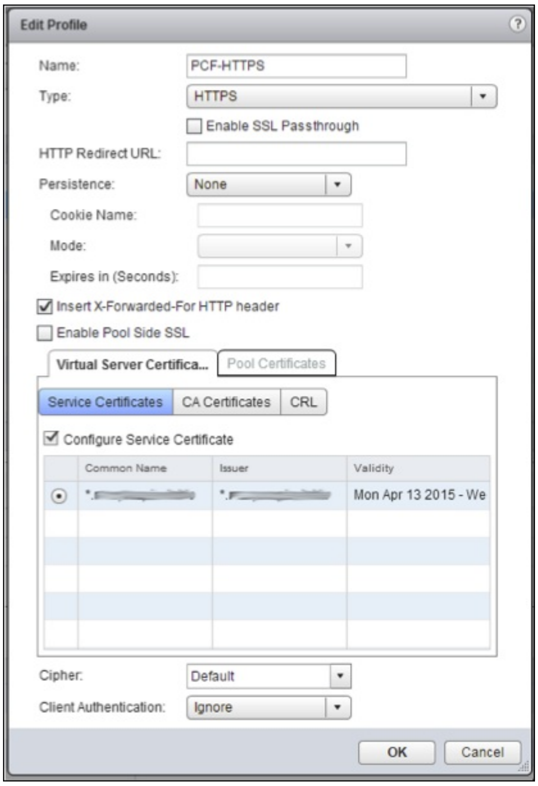
To create the application profiles, access the ESG UI and perform the following steps:

1. Select **Edge**, **Manage**, **Load Balancer**, and then **Global Application Profiles**.

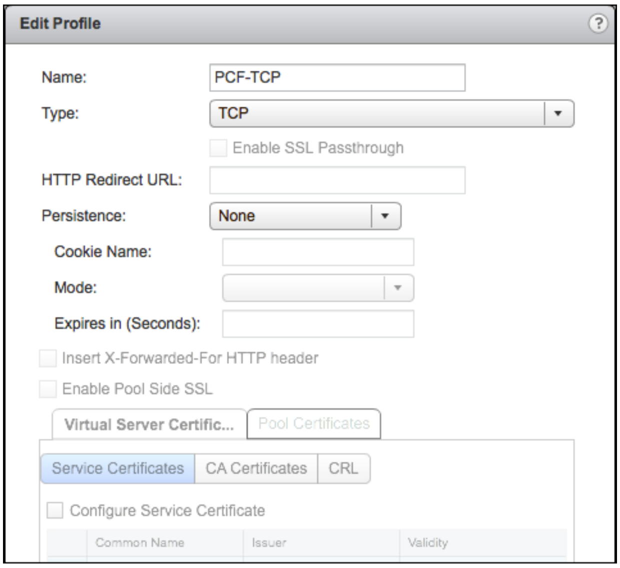
2. Create/Edit Profile and make the **PCF-HTTP** rule, turning on **Insert X-Forwarded-For HTTP header**.



3. Create/Edit Profile and make the **PCF-HTTPS** rule, same as before, but add the service certificate inserted before.



4. Create/Edit Profile and make **PCF-TCP** rule, with the Type set to TCP.



Step 2.4: Create Application Rules

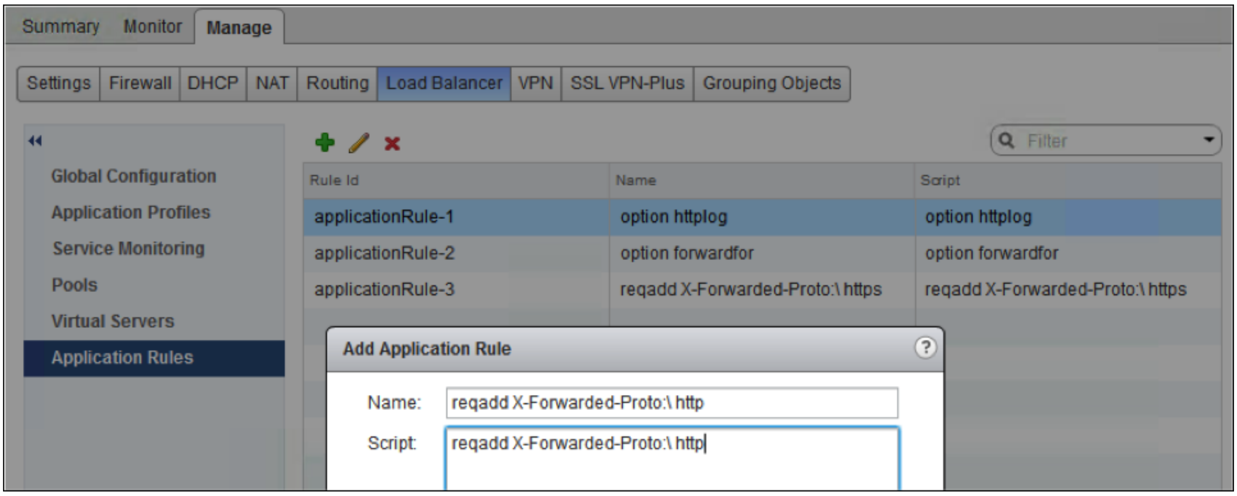
In order for the ESG to perform proper `x-forwarded` requests, you need to add a few HAProxy directives to the ESG Application Rules. NSX supports most directives that HAProxy supports.

To create the application rules, access the ESG UI and perform the following steps:

1. Select **Edge**, **Manage**, **Load Balancer**, and then **Application Rules**.

2. Copy and paste the table entries below into each field, one per rule.

Rule Name	Script
option httplog	option httplog
reqadd X-Forwarded-Proto:\ https	reqadd X-Forwarded-Proto:\ https
reqadd X-Forwarded-Proto:\ http	reqadd X-Forwarded-Proto:\ http



Step 2.5: Create Monitors For Pools

NSX ships with several load balancing monitoring types pre-defined. These types are for HTTP, HTTPS and TCP. For this installation, operators build new monitors matching the needs of each pool to ensure correct 1:1 monitoring for each pool type.

To create monitors for pools, access the ESG UI and perform the following steps:

1. Select **Edge**, **Manage**, **Load Balancer**, and then **Service Monitoring**.

2. Create a new monitor for `http-routers` , and keep the defaults.
3. Set the Type to `HTTP` .
4. Set the Method to `GET` .
5. Set the URL to `/health` .
6. Create a new monitor for `tcp-routers` , and keep the defaults.
7. Set the type to `HTTP` .
8. Set the Method to `GET` .
9. Set the URL to `/health` .
10. Create a new monitor for `diego-brains` , and keep the defaults.
11. Set the type to `TCP` .
12. Create a new monitor for `ert-mysql-proxy` , and keep the defaults.
13. Set the type to `TCP` .

These monitors are selected during the next step when pools are created. A pool and a monitor are matched 1:1.

Step 2.6: Create Pools of Multi-Element PCF Targets

The following steps creates the pools of resources that ESG is load balancing * **TO** *, which are the Gorouter, TCP Router, Diego Brain, and PAS MySQL Proxy jobs deployed by BOSH Director. If the IP addresses specified in the configuration do not exactly match the IP addresses reserved or used for the resources, then the pool will not effectively load balance.

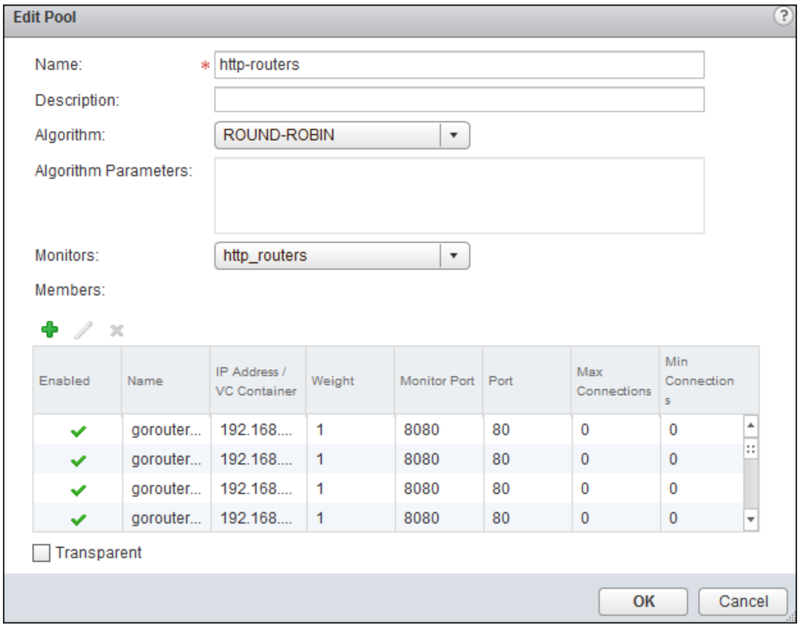
Step 2.6a: Create Pool for `http-routers`

To create pool for `http-routers` , access the ESG UI and perform the following steps:

1. Select **Edge**, **Manage**, **Load Balancer**, and then **Pools**.
2. Enter ALL the IP addresses reserved for the Gorouters into this pool. If you reserved more addresses than you have Gorouters, enter the addresses anyway and the load balancer ignores the missing resources as “down”.

   > **Note**: If your deployment matches the Reference Architecture for PCF on vSphere, these IP addresses are in the 192.168.20.0/22 address space.

3. If required, adjust **Port** and **Monitor Port**. Note that by default the port and monitoring port are on HTTP port 80. The assumption is that internal traffic from the ESG load balancer to the Gorouters is trusted because it is on a VXLAN secured within NSX. If using encrypted traffic inside the load balancer, adjust the ports accordingly.
4. Set the **Algorithm** to `ROUND-ROBIN` .



5. Set **Monitors** to `http-routers`.

Step 2.6b: Create Pool for `tcp-routers`

1. Select **Edge**, **Manage**, **Load Balancer**, and then **Pools**.

2. Enter ALL the IP addresses reserved for TCP Routers into this pool. If you reserved more addresses than you have VMs, enter the addresses anyway and the load balancer ignores the missing resources as “down”.

    > **Note:** If your deployment matches the Reference Architecture for PCF on vSphere, these IP addresses are in the 192.168.20.0/22 address space.

3. Set the **Port** to empty (these numbers vary) and the **Monitor Port** to 80.

4. Set the **Algorithm** to `ROUND-ROBIN`.

5. Set the **Monitors** to `tcp-routers`.

Step 2.6c: Create Pool for `diego-brains`

1. Select **Edge**, **Manage**, **Load Balancer**, and then **Pools**.

2. Enter ALL the IP addresses reserved for Diego Brains into this pool. If you reserved more addresses than you have VMs, enter the addresses anyway and the load balancer will just ignore the missing resources as “down”.

    > **Note:** If your deployment matches the Reference Architecture for PCF on vSphere, these IP addresses are in the 192.168.20.0/22 address space.

3. Set the **Port** to 2222 and the **Monitor Port** to 2222.

4. Set the Algorithm to `ROUND-ROBIN`.

5. Set the Monitors to `diego-brains`.

Step 2.6d: Create Pool for `ert-mysql-proxy`

1. Select **Edge**, **Manage**, **Load Balancer**, and then **Pools**.

2. Enter the two IP addresses reserved for MySQL-proxy into this pool.

    > **Note:** If your deployment matches the Reference Architecture for PCF on vSphere, these IP addresses are in the 192.168.20.0/22 address

space.

3. Set the **Port** to 3306 and the **Monitor Port** to 1936.

4. Set the **Algorithm** to `ROUND-ROBIN`.

5. Set the **Monitors** to `ert-mysql-proxies`.

Step 2.7: Create Virtual Servers

This is the Virtual IP (VIP) that the load balancer uses to represent the pool of Gorouters to the outside world. This also links the Application Policy, Application Rules, and backend pools to provide PCF load balancing services. This is the interface that the load balancer balances **from**. You create three Virtual Servers.

1. Select **Edge**, **Manage**, **Load Balancer**, and then **Virtual Servers**.

2. Select an IP address from the available routable address space allocated to the ESG. For information about reserved IP addresses, see General Overview.

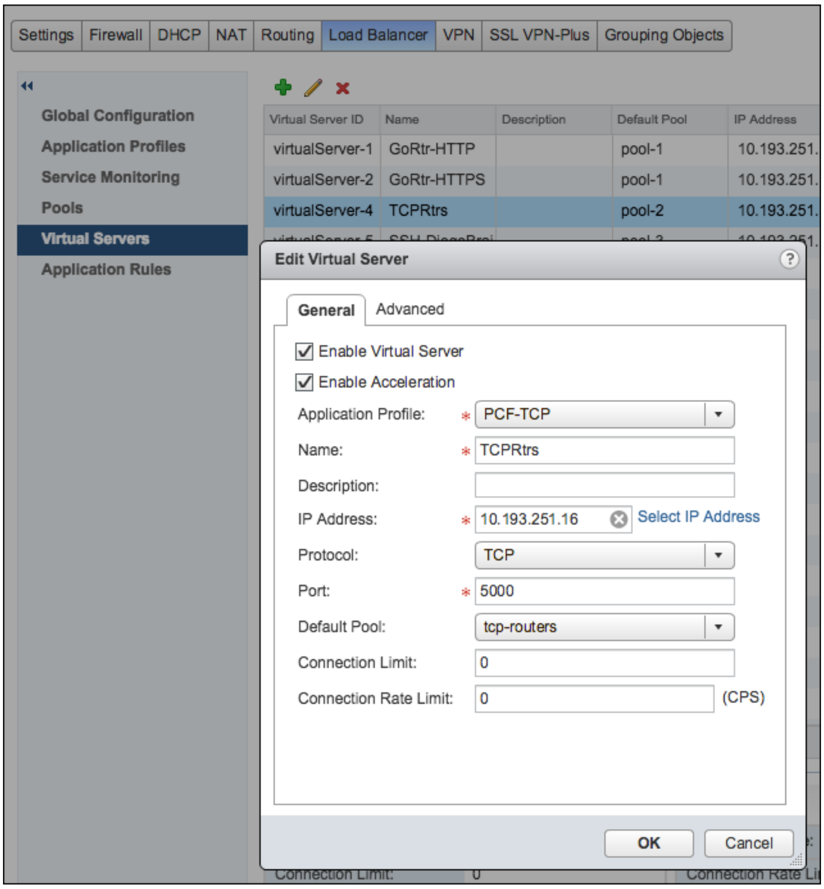
3. Create a new Virtual Server named `GoRtr-HTTP` and select Application Profile `PCF-HTTP`.
   - Use **Select IP Address** to select the IP address to use as a VIP on the uplink interface.
   - Set **Protocol** to match the **Application Profile** protocol (HTTP) and set **Port** to match the protocol (80).
   - Set **Default Pool** to the pool name set in the above procedure. This connects this VIP to the pool of resources being balanced to.
   - Ignore **Connection Limit** and **Connection Rate Limit** unless these limits are desired.
   - Switch to **Advanced Tab** on this Virtual Server.
   - Use the green plus to add/attach three Application Rules to this Virtual Server:
     - option httplog
     - reqadd X-Forwarded-Proto:\ http

     > **Note**: Be careful to match protocol rules to the protocol `VIP- HTTP` to `HTTP` and `HTTPS` to `HTTPS`.

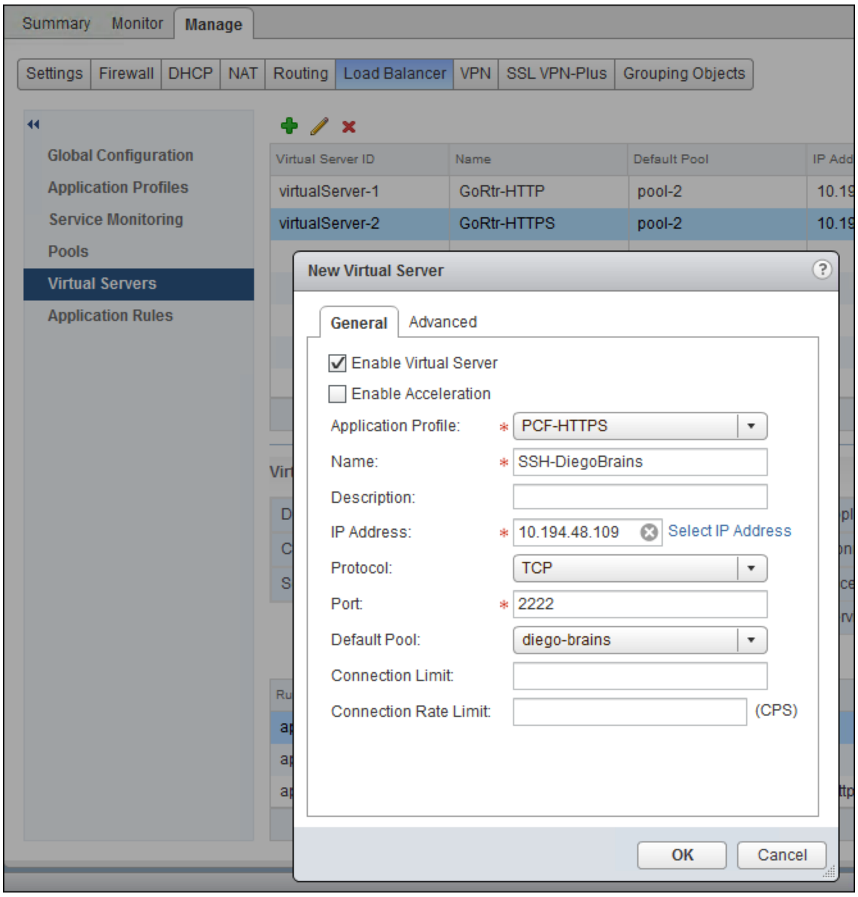
4. Create a new Virtual Server named `GoRtr-HTTPS` and select Application Profile `PCF-HTTPS`.
   - Use **Select IP Address** to select the **same IP address** to use as a VIP on the uplink interface.
   - Set **Protocol** to match the **Application Profile** protocol (`HTTPS`) and set **Port** to match the protocol (443).
   - Set **Default Pool** to the pool name set in the above procedure (`http-routers`). This connects this VIP to that pool of resources being balanced to.
   - Ignore **Connection Limit** and **Connection Rate Limit** unless these limits are desired.
   - Switch to **Advanced Tab** on this Virtual Server.
   - Use the green plus to add/attach three Application Rules to this Virtual Server:
     - option httplog
     - reqadd X-Forwarded-Proto:\ https

     > **Note**: Be careful to match protocol rules to the protocol `VIP- HTTP` to `HTTP` and `HTTPS` to `HTTPS`.

5. Create a new Virtual Server named `TCPRtrs` and select Application Profile `PCF-TCP`.
   - Use **Select IP Address** to select the IP address to use as a VIP on the uplink interface.
   - Set **Protocol** to match the **Application Profile** protocol (TCP) and set **Port** to match the protocol (5000).
   - Set **Default Pool** to the pool name set in the above procedure (`tcp-routers`). This connects this VIP to the pool of resources being balanced to.
   - Ignore **Connection Limit** and **Connection Rate Limit** unless these limits are desired.



6. Create a new Virtual Server named `SSH-DiegoBrains` and select Application Profile `PCF-HTTPS`.
   - Use **Select IP Address** to select the same IP address to use as a VIP on the uplink interface if you want to use this address for SSH access to apps. If not, select a different IP address to use as the VIP.
   - Set **Protocol** to TCP and set **Port** to 2222.
   - Set **Default Pool** to the pool name set in the above procedure (`diego-brains`). This connects this VIP to that pool of resources being balanced to.
   - Ignore **Connection Limit** and **Connection Rate Limit** unless these limits are desired.



Step 3: Configure NAT/SNAT

The ESG obfuscates the PCF installation through network translation. The PCF installation is placed entirely on non-routable RFC-1918 network address space, so to be useful, you must translate routable IP addresses to non-routable IP addresses to make connections.

Note: Correct NAT/SNAT configuration is required for the PCF installation to function correctly.

Action	Applied on Interface	Original IP	Original Port	Translated IP	Translated Port	Protocol	Description
SNAT	uplink	192.168.0.0/16	any	IP_of_PCF	any	any	All Nets Egress
DNAT	uplink	IP_of_OpsMgr	any	192.168.10.OpsMgr	any	tcp	OpsMgr Mask

NAT/SNAT functionality is not required if routable IP address space is used on the Tenant Side of the ESG. At that point, the ESG simply performs routing between the address segments.

Note: NSX will generate a number of DNAT rules based on load balancing configs. These can safely be ignored.

Additional Notes

The ESG also supports scenarios where Private RFC subnets and NAT are not utilized for **Deployment** or **Infrastructure** networks, and the guidance in this document can be modified to meet those scenarios.

Additionally, the ESG supports up to 10 Interfaces allowing for more Uplink options if necessary.

The use of Private RFC-1918 subnets for PCF Deployment networks was chosen due to its popularity with customers. ESG devices are capable of leveraging ECMP, OSPF, BGP, and IS-IS to handle dynamic routing of customer and/or public L3 IP space. That design is out of scope for this document, but is supported by VMware NSX and Pivotal PCF.

Upgrading vSphere without PCF Downtime

Page last updated:

This topic describes how to upgrade the vSphere components that host your Pivotal Cloud Foundry (PCF) installation without service disruption.

Minimum Requirements

At a bare minimum, vSphere contains the following components:

- vCenter Server
- one or more ESXi hosts

You cannot perform an in-place upgrade of vSphere without at least two ESXi hosts in your cluster.

If you do not meet this requirement (in other words, you have insufficient resources to evacuate an entire host), then you may experience PCF downtime during the upgrade.

To upgrade vSphere with only one ESXi host or without sufficient headroom capacity, you must reduce your PCF installation size. In other words, you can either reduce the number of Diego cells in your deployment or pause PCF VMs to make more capacity available. These actions can result in PCF downtime.

Recommended Starting Configuration

If you are running a PCF deployment as recommended by the base reference architecture for PCF on vSphere (recommended), then your vSphere installation should have the following components:

- One vCenter Server
- Three ESXi hosts per cluster
- Three or more clusters
- One (or HA pair) NSX Edge appliances

> **Note**: Pivotal recommends having at least three ESXi hosts in your cluster to maintain PCF high availability during your upgrade.

For more information, see the Reference Architecture for Pivotal Cloud Foundry on vSphere.

Procedure to Upgrade vSphere

To upgrade the vSphere management layer underneath PCF, perform the following steps:

Step 1. Upgrade vCenter

For example, you might be upgrading vCenter 6.0 to vCenter 6.5.

For more information about how to upgrade vCenter, see Overview of the vCenter Server Upgrade Process in VMware documentation.

Step 2. Upgrade ESXi Hosts

After a successful vCenter upgrade, upgrade your ESXi hosts one at a time.

Starting with the first ESXi host, perform the following steps:

1. Verify that your ESXi hosts have sufficient resources and headroom to evacuate the VM workload of a single ESXi host to the two remaining hosts.

   > **Note**: If you have enabled vSphere HA on your ESXi host, then each ESXi host should have sufficient headroom capacity since HA reserves 66% of available memory.

2. Use vMotion to move all the PCF VMs on the host you want to upgrade to the other ESXi hosts. vMotion places the VMs on the other hosts based on available capacity. For more information, see Migration with vMotion in VMware documentation.

3. Upgrade the evacuated ESXi host. For example, you may be upgrading from ESXi v6.0 to ESX v6.5. For instructions, see Upgrading ESXi Hosts in VMware documentation.

After successfully upgrading the first ESXi host, repeat the above steps for each remaining host one at a time. vSphere automatically rebalances all PCF VMs back onto the upgraded hosts via DRS after all the hosts are done.

Step 3. Upgrade ESG on VMware NSX

If your PCF deployment lives on a network behind an Edge Services Gateway (ESG) as recommended by the reference architecture, then upgrade each ESG only after completing the upgrade of vCenter and your ESXi hosts.

When you upgrade an ESG on VMware NSX, you upgrade the NSX Manager software. This upgrade can cause some slight downtime, the amount of which depends on the number of ESGs you are using.

- If your deployment only has one ESG, you can expect a downtime of 5 minutes for network reconvergence.
- If your ESGs are deployed in HA, upgrade the first ESG. Then upgrade the second ESG. This upgrade results only in 15-20 seconds of downtime.

For more information, see the NSX Upgrade Guide in VMware documentation.

Migrating PCF to a New Datastore in vSphere

Page last updated:

This topic describes how to migrate your Pivotal Cloud Foundry (PCF) installation to a new vSphere datastore.

Prerequisites

Both the new and existing vSphere datastores must reside in the same datacenter.

To avoid service disruption, Pivotal recommends that you configure your overall PCF deployment for high availability. In addition, check for configurations necessary to achieve high availability in each of your installed product tiles.

If your environment has any single points of failure, service may be disrupted as a result of the migration.

Before You Begin

This section describes the steps you should perform prior to the migration.

Step 1: Backup Your Environment

Ensure that your PCF environment is fully backed up.

For more information about how to backup PCF, see Backing up Pivotal Cloud Foundry.

Step 2: Document Current Environment Settings

Document your current environment settings before proceeding with the datastore migration. Record which VMs are running and in which datastore they reside. If you experience any issues during or after the migration, you must have this information to restore your environment.

To obtain this information, perform the following steps:

1. Run the `bosh instances` command.
    - If you use BOSH CLI v2, run the following command, replacing `MY-ENV` with the alias you assigned to your BOSH Director:

      ```
      $ bosh -e MY-ENV instances --details > instances.txt
      ```

    - If you use the original version of the BOSH CLI, run the following command:

      ```
      $ bosh-old instances --details > instances.txt
      ```

2. Save the resulting file `instances.txt` to a safe location.

3. Note the datastore where each VM resides in vSphere.

Step 3: Modify CPI Timeout Value (Optional)

The default timeout for the BOSH CPI is 60 minutes. When performing a datastore migration, BOSH must copy all of the data from the old disks to the new disks within this time limit. In general, most copy operations should fit within this time limit, but it ultimately depends on the hardware in your deployment and the size of your existing persistent disks.

To determine whether 60 minutes is sufficient for the datastore migration, estimate how long it takes to copy 100 GB of data. Then, based on the size of your persistent disks, determine whether 60 minutes is sufficient time to copy that amount of data.

If you have previously encountered `out of sync` errors when modifying your PCF deployment, you should increase the timeout value of the CPI before migrating the datastores.

To modify the default BOSH CPI timeout, follow the instructions in the following KB article.

- How to increase timeout on BOSH CPI command calls

For more information about resolving the error after migrating your datastore, see After the Migration.

Step 4: Check System Health

In Pivotal Application Service (PAS), check the **Status** tab and make sure there are no errors or reported issues.

Step 5: Check Installed Products Health

In each tile installed in your PCF deployment, check the **Status** tab and make sure there are no errors or reported issues.

Step 6: Check Ops Manager Director Status

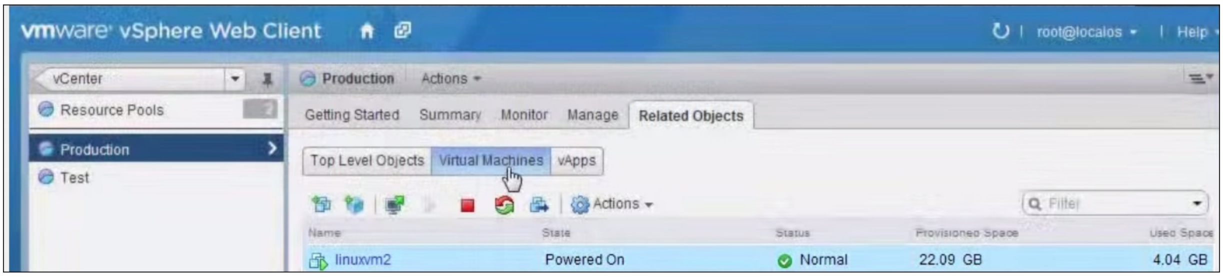
1. Check that there are no pending changes and that the status of all tiles is green.
2. Make sure the last Installation Log does not contain any errors.
3. Before proceeding with the migration, click **Apply Changes** to make sure there are no errors in the Installation Log.

Procedure: Migrate PCF to a New Datastore

1. In Ops Manager Director, navigate to the **vCenter Config** page.
2. Update the **Ephemeral Datastore Names** and **Persistent Datastore Names** field to reflect the new datastore names, then click **Save**.

   > **Note**: If you use the Datastore Clustering feature in vSphere, provide only the individual names of the datastores in the cluster. Do not provide the name of the cluster that contains them.

3. Click **Apply Changes**.
4. Confirm that the Ops Manager Director VM has persistent disk on the new datastore.
   a. Navigate to **vCenter Resource Pools** and select the **Resource Pool** that contains your PCF deployment VMs and new datastore.
   b. Click the **Related Objects** and **Virtual Machines**.
   c. Locate the Ops Manager VM and verify that the VM has an expected value in the **Provisioned Space** column.



5. In Ops Manager Director, navigate to the **Director Config** page, and select the **Recreate all VMs** option.
6. Click **Apply Changes**.

After the Migration

When BOSH moves disks, it waits for up to 60 minutes for the operation to complete. If the operation does not complete in time, BOSH can enter a state where it claims that the disks are `out of sync`.

Fix Failed BOSH Deployment with Out-of-Sync Error

If your PCF deployment gets into this state, you can resolve the issue by performing the steps in the following KB article:

- How to recover from a failed bosh deployment when VMs are out of sync on vSphere

Prevent Out-of-Sync Error

You can also prevent the `out of sync` BOSH error by increasing the CPI timeout to a larger value before performing the migration. Follow the instructions in the following KB article.

- How to increase timeout on Bosh CPI command calls

PCF Dev Overview

Page last updated:

This guide describes how to install and use PCF Dev, a lightweight Pivotal Cloud Foundry (PCF) installation that runs on a single virtual machine (VM) on your workstation. PCF Dev is intended for application developers who want to develop and debug their applications locally on a PCF deployment.

PCF Dev includes Pivotal Application Service (PAS) , Redis, RabbitMQ, and MySQL. It also supports all Cloud Foundry Command Line Interface (cf CLI) functionality. See the Comparing PCF Dev to Pivotal Cloud Foundry table below for more product details.

Prerequisites

- VirtualBox: 5.0+: PCF Dev uses VirtualBox as its virtualizer.
- The latest version of the cf CLI: Use the cf CLI to push and scale apps.
- You must have an Internet connection for DNS. See Using PCF Dev Offline if you do not have an Internet connection.
- At least 3 GB of available memory on your host machine. Pivotal recommends running on a host system with at least 8 GB of total RAM.

Installing PCF Dev

- Installing PCF Dev on Mac OS X
- Installing PCF Dev on Linux
- Installing PCF Dev on Microsoft Windows

Configuring and Using PCF Dev

- Configuring PCF Dev
- Using PCF Dev
- Using Services in PCF Dev
- Using Spring Cloud Services in PCF Dev
- Using PCF Dev Behind a Proxy
- Using PCF Dev Offline
- PCF Dev on AWS
- Frequently Asked Questions

Comparing PCF Dev to Pivotal Cloud Foundry

PCF Dev mirrors PCF in its key product offerings. If an application runs on PCF Dev, it runs on PCF with no modification in almost all cases. Review the table below for key product details.

	PCF Dev	PCF	CF
Space required	20 GB	100GB+	50GB+
Memory required	3 GB	50GB+	variable
Deployment	`cf dev start`	Ops Manager	`bosh create-env`
Estimated time-to-deploy	10 Minutes	Hour+	Hour+
Out-of-the-Box Services	Redis MySQL RabbitMQ	Redis MySQL RabbitMQ GemFire	N/A
PAS	✓	✓	✓
Logging/Metrics	✓	✓	✓
Routing	✓	✓	✓

	PCF Dev	PCF	CF
Routing	✓	✓	✓
Compatible with CF CLI	✓	✓	✓
Deploy apps with any supported buildpack	✓	✓	✓
Supports Multi-Tenancy	✓	✓	✓
Diego Support	✓	✓	✓
Docker Support	✓	✓	✓
User-Provided Services	✓	✓	✓
High Availability		✓	✓
Integration with 3rd party Authorization		✓	✓
BOSH Director (i.e., can perform additional BOSH deployments)		✓	✓
Day Two Lifecycle Operations (e.g., rolling upgrades, security patches)		✓	✓
Ops Manager		✓	
Apps Manager	✓	✓	
Tile Support		✓	
Developers have root-level access across cluster	✓		
Pre-provisioned	✓		
Does not depend on BOSH	✓		

Monitoring Pivotal Cloud Foundry

This guide describes how Pivotal Cloud Foundry (PCF) operators can monitor their deployments.

In This Guide

This guide includes the following topics:

- Key Performance Indicators: A list of Key Performance Indicators (KPIs) that operators may want to monitor with their PCF deployment to help ensure it is in a good operational state.
- Key Capacity Scaling Indicators: A list of capacity scaling indicators that operators may want to monitor to determine when they need to scale their PCF deployments.
- Configuring a Monitoring System: Guidance for setting up PCF with third-party monitoring platforms to continuously monitor system metrics and trigger health alerts.

For information about logging and metrics in PCF and about monitoring of services for PCF, see Additional Resources below.

KPI Changes from PCF v1.12 to v2.0

This table highlights new and changed KPIs in PCF v2.0.

Modified	KPI: `Adapter Loss Rate` The origin name of these metrics has changed from `scalablesyslog` to `cf-syslog-drain`.	Link
Modified	KPI: `Syslog Drain Bindings Count` The origin name of this metric has changed from `scalablesyslog` to `cf-syslog-drain`.	Link
New	KPI: `bbs.LockHeld` New capability to monitor Diego active locks at the BBS component level.	Link
New	KPI: `auctioneer.LockHeld` New capability to monitor Diego active locks at the Auctioneer component level.	Link
New	KPI: `Number of Route Registration Messages Sent and Received` New capability to monitor for issues in the control plane responsible for updating the routers with changes to the routing table. This recommendation was also added to the PCF 1.12, 1.11, and 1.10 versions of the Key Performance Indicators documentation.	Link

Additional Resources

For information about logging and metrics in PCF, see the following topics:

- Configuring System Logging in PAS: This topic explains how to configure the PCF Loggregator system to scale its maximum throughput and to forward logs to an external aggregator service.
- Logging and Metrics: A guide to Loggregator, the system which aggregates and streams logs and metrics from user apps and system components in PAS.

For information about KPIs and metrics for PCF services, see the following topics:

- **RabbitMQ for PCF (on-demand):** Monitoring and KPIs for On-Demand RabbitMQ for PCF
- **RabbitMQ for PCF (pre-provisioned):** Monitoring and KPIs for Pre-Provisioned RabbitMQ for PCF

Key Performance Indicators

This topic describes Key Performance Indicators (KPIs) that operators may want to monitor with their Pivotal Cloud Foundry (PCF) deployment to help ensure it is in a good operational state.

The following PCF v1.12 KPIs are provided for operators to give general guidance on monitoring a PCF deployment using platform component and system (BOSH) metrics. Although many metrics are emitted from the platform, the following PCF v1.12 KPIs are high-signal-value metrics that can indicate emerging platform issues.

This alerting and response guidance has been shown to apply to most deployments. Pivotal recommends that operators continue to fine-tune the alert measures to their deployment by observing historical trends. Pivotal also recommends that operators expand beyond this guidance and create new, deployment-specific monitoring metrics, thresholds, and alerts based on learning from their deployments.

Note: Thresholds noted as "dynamic" in the tables below indicate that while a metric is highly important to watch, the relative numbers to set threshold warnings at are specific to a given PCF environment and its use cases. These dynamic thresholds should be occasionally revisited because the PCF foundation and its usage continue to evolve. See Determine Warning and Critical Thresholds for more information.

Diego Auctioneer Metrics

Auctioneer App Instance (AI) Placement Failures

auctioneer.AuctioneerLRPAuctionsFailed	
Description	The number of Long Running Process (LRP) instances that the auctioneer failed to place on Diego cells. This metric is cumulative over the lifetime of the auctioneer job. **Use**: This metric can indicate that PCF is out of container space or that there is a lack of resources within your environment. This indicator also increases when the LRP is requesting an isolation segment, volume drivers, or a stack that is unavailable, either not deployed or lacking sufficient resources to accept the work. This error is most common due to capacity issues, for example, if cells do not have enough resources, or if cells are going back and forth between a healthy and unhealthy state. **Origin**: Firehose **Type**: Counter (Integer) **Frequency**: During each auction
Recommended measurement	Per minute delta averaged over a 5-minute window
Recommended alert thresholds	**Yellow warning**: ≥ 0.5 **Red critical**: ≥ 1
Recommended response	1. To best determine the root cause, examine the Auctioneer logs. Depending on the specific error and resource constraint, you may also find a failure reason in the Cloud Controller (CC) API. 2. Investigate the health of your Diego cells to determine if they are the resource type causing the problem. 3. Consider scaling additional cells using Ops Manager. 4. If scaling cells does not solve the problem, pull Diego brain logs and BBS node logs and contact Pivotal Support telling them that LRP auctions are failing.

Auctioneer Time to Fetch Cell State

auctioneer.AuctioneerFetchStatesDuration	
	Time in ns that the auctioneer took to fetch state from all the Diego cells when running its auction.

Description	**Use**: Indicates how the cells themselves are performing. Alerting on this metric helps alert that app staging requests to Diego may be failing. **Origin**: Firehose **Type**: Gauge, integer in ns **Frequency**: During event, during each auction
Recommended measurement	Maximum over the last 5 minutes divided by 1,000,000,000
Recommended alert thresholds	**Yellow warning**: ≥ 2 s **Red critical**: ≥ 5 s
Recommended response	1. Check the health of the cells by reviewing the logs and looking for errors. 2. Review IaaS console metrics. 3. Pull Diego brain logs and cell logs and contact Pivotal Support telling them that fetching cell states is taking too long.

Auctioneer App Instance Starts

auctioneer.AuctioneerLRPAuctionsStarted	
Description	The number of LRP instances that the auctioneer successfully placed on Diego cells. This metric is cumulative over the lifetime of the auctioneer job. **Use**: Provides a sense of running system activity levels in your environment. Can also give you a sense of how many app instances have been started over time. The recommended measurement, below, can help indicate a significant amount of container churn. However, for capacity planning purposes, it is more helpful to observe deltas over a long time window. **Origin**: Firehose **Type**: Counter (Integer) **Frequency**: During event, during each auction
Recommended measurement	Per minute delta averaged over a 5-minute window
Recommended alert thresholds	**Yellow warning**: Dynamic **Red critical**: Dynamic
Recommended response	When observing a significant amount of container churn, do the following: 1. Look to eliminate explainable causes of temporary churn, such as a deployment or increased developer activity. 2. If container churn appears to continue over an extended period, pull logs from the Diego Brain and BBS node before contacting Pivotal support. When observing extended periods of high or low activity trends, scale up or down CF components as needed.

Auctioneer Task Placement Failures

auctioneer.AuctioneerTaskAuctionsFailed	
Description	The number of Tasks that the auctioneer failed to place on Diego cells. This metric is cumulative over the lifetime of the auctioneer job. **Use**: Failing Task auctions indicate a lack of resources within your environment and that you likely need to scale. This indicator also increases when the Task is requesting an isolation segment, volume drivers, or a stack that is unavailable, either not deployed or lacking sufficient resources to accept the work. This error is most common due to capacity issues, for example, if cells do not have enough resources, or if cells are going back and forth between a healthy and unhealthy state.

	Origin: Firehose **Type**: Counter (Float) **Frequency**: During event, during each auction
Recommended measurement	Per minute delta averaged over a 5-minute window
Recommended alert thresholds	**Yellow warning**: ≥ 0.5 **Red critical**: ≥ 1
Recommended response	1. In order to best determine the root cause, examine the Auctioneer logs. Depending on the specific error or resource constraint, you may also find a failure reason in the CC API. 2. Investigate the health of Diego cells. 3. Consider scaling additional cells using Ops Manager. 4. If scaling cells does not solve the problem, pull Diego brain logs and BBS logs for troubleshooting and contact Pivotal Support for additional troubleshooting. Inform Pivotal Support that Task auctions are failing.

Diego BBS Metrics

BBS Time to Run LRP Convergence

bbs.ConvergenceLRPDuration	
Description	Time in ns that the BBS took to run its LRP convergence pass. **Use**: If the convergence run begins taking too long, apps or Tasks may be crashing without restarting. This symptom can also indicate loss of connectivity to the BBS database. **Origin**: Firehose **Type**: Gauge (Integer in ns) **Frequency**: During event, every 30 seconds when LRP convergence runs, emission should be near-constant on a running deployment
Recommended measurement	Maximum over the last 15 minutes divided by 1,000,000,000
Recommended alert thresholds	**Yellow warning**: ≥ 10 s **Red critical**: ≥ 20 s
Recommended response	1. Check BBS logs for errors. 2. Try vertically scaling the BBS VM resources up. For example, add more CPUs or memory depending on its `system.cpu` / `system.memory` metrics. 3. If that does not solve the issue, pull the BBS logs and contact Pivotal Support for additional troubleshooting.

BBS Time to Handle Requests

bbs.RequestLatency	
Description	The maximum observed latency time over the past 60 seconds that the BBS took to handle requests across all its API endpoints. Diego is now aggregating this metric to emit the max value observed over 60 seconds. **Use**: If this metric rises, the PCF API is slowing. Response to certain cf CLI commands is slow if request latency is high.

	...gh... **Origin**: Firehose **Type**: Gauge (Integer in ns) **Frequency**: 60 s
Recommended measurement	Average over the last 15 minutes divided by 1,000,000,000
Recommended alert thresholds	**Yellow warning**: ≥ 5 s **Red critical**: ≥ 10 s
Recommended response	1. Check CPU and memory statistics in Ops Manager. 2. Check BBS logs for faults and errors that can indicate issues with BBS. 3. Try scaling the BBS VM resources up. For example, add more CPUs/memory depending on its `system.cpu` / `system.memory` metrics. 4. If the above steps do not solve the issue, collect a sample of the cell logs from the BBS VMs and contact Pivotal Support to troubleshoot further.

Cloud Controller and Diego in Sync

bbs.Domain.cf-apps	
Description	Indicates if the `cf-apps` Domain is up-to-date, meaning that CF App requests from Cloud Controller are synchronized to `bbs.LRPsDesired` (Diego-desired AIs) for execution. • `1` means `cf-apps` Domain is up-to-date • No data received means `cf-apps` Domain is not up-to-date **Use**: If the `cf-apps` Domain does not stay up-to-date, changes requested in the Cloud Controller are not guaranteed to propagate throughout the system. If the Cloud Controller and Diego are out of sync, then apps running could vary from those desired. **Origin**: Firehose **Type**: Gauge (Float) **Frequency**: 30 s
Recommended measurement	Average over the last 5 minutes
Recommended alert thresholds	**Yellow warning**: *N/A* **Red critical**: < 1
Recommended response	1. Check the BBS logs. 2. If the problem continues, pull Diego brain logs and BBS logs and contact Pivotal Support to say that the `cf-apps` domain is not being kept fresh.

More App Instances Than Expected

bbs.LRPsExtra	
Description	Total number of LRP instances that are no longer desired but still have a BBS record. When Diego wants to add more apps, the BBS sends a request to the auctioneer to spin up additional LRPs. LRPsExtra is the total number of LRP instances that are no longer desired but still have a BBS record. **Use**: If Diego has more LRPs running than expected, there may be problems with the BBS. Deleting an app with many instances can temporarily spike this metric. However, a sustained spike in `bbs.LRPsExtra` is unusual and should be investigated. **Origin**: Firehose **Type**: Gauge (Float)

	Frequency: 30 s
Recommended measurement	Average over the last 5 minutes
Recommended alert thresholds	**Yellow warning**: ≥ 5 **Red critical**: ≥ 10
Recommended response	1. Review the BBS logs for proper operation or errors, looking for detailed error messages. 2. If the condition persists, pull the BBS logs and contact Pivotal Support.

Fewer App Instances Than Expected

bbs.LRPsMissing	
Description	Total number of LRP instances that are desired but have no record in the BBS. When Diego wants to add more apps, the BBS sends a request to the auctioneer to spin up additional LRPs. LRPsMissing is the total number of LRP instances that are desired but have no BBS record. **Use**: If Diego has less LRP running than expected, there may be problems with the BBS. An app push with many instances can temporarily spike this metric. However, a sustained spike in `bbs.LRPsMissing` is unusual and should be investigated. **Origin**: Firehose **Type**: Gauge (Float) **Frequency**: 30 s
Recommended measurement	Average over the last 5 minutes
Recommended alert thresholds	**Yellow warning**: ≥ 5 **Red critical**: ≥ 10
Recommended response	1. Review the BBS logs for proper operation or errors, looking for detailed error messages. 2. If the condition persists, pull the BBS logs and contact Pivotal Support.

Crashed App Instances

bbs.CrashedActualLRPs	
Description	Total number of LRP instances that have crashed. **Use**: Indicates how many instances in the deployment are in a crashed state. An increase in `bbs.CrashedActualLRPs` can indicate several problems, from a bad app with many instances associated, to a platform issue that is resulting in app crashes. Use this metric to help create a baseline for your deployment. After you have a baseline, you can create a deployment-specific alert to notify of a spike in crashes above the trend line. Tune alert values to your deployment. **Origin**: Firehose **Type**: Gauge (Float) **Frequency**: 30 s
Recommended measurement	Average over the last 5 minutes
Recommended alert thresholds	**Yellow warning**: Dynamic **Red critical**: Dynamic
Recommended response	1. Look at the BBS logs for apps that are crashing and at the cell logs to see if the problem is with the apps themselves, rather than a platform issue. 2. Before contacting Pivotal Support, pull the BBS logs and, if particular apps are the problem, pull the logs from their Diego cells too.

Running App Instances, Rate of Change

	1hr average of bbs.LRPsRunning – prior 1hr average of bbs.LRPsRunning
Description	Rate of change in app instances being started or stopped on the platform. It is derived from `bbs.LRPsRunning` and represents the total number of LRP instances that are running on Diego cells. **Use**: Delta reflects upward or downward trend for app instances started or stopped. Helps to provide a picture of the overall growth trend of the environment for capacity planning. You may want to alert on delta values outside of the expected range. **Origin**: Firehose **Type**: Gauge (Float) **Frequency**: During event, emission should be constant on a running deployment
Recommended measurement	derived=(1-hour average of `bbs.LRPsRunning` – prior 1-hour average of `bbs.LRPsRunning`)
Recommended alert thresholds	**Yellow warning**: Dynamic **Red critical**: Dynamic
Recommended response	Scale components as necessary.

Diego Cell Metrics

Remaining Memory Available — Cell Memory Chunks Available

	rep.CapacityRemainingMemory
Description	Remaining amount of memory in MiB available for this Diego cell to allocate to containers. **Use**: Indicates the available cell memory. Insufficient cell memory can prevent pushing and scaling apps. The strongest operational value of this metric is to understand a deployment's average app size and monitor/alert on ensuring that at least some cells have large enough capacity to accept standard app size pushes. For example, if pushing a 4GB app, Diego would have trouble placing that app if there is no one cell with sufficient capacity of 4GB or greater. As an example, Pivotal Cloud Ops uses a standard of 4GB, and computes and monitors for the number of cells with at least 4GB free. When the number of cells with at least 4GB falls below a defined threshold, this is a scaling indicator alert to increase capacity. This *free chunk* count threshold should be tuned to the deployment size and the standard size of apps being pushed to the deployment. **Origin**: Firehose **Type**: Gauge (Integer in bytes) **Frequency**: 60 s
Recommended measurement	**For alerting**: 1. Determine the size of a standard app in your deployment. This is the suggested value to calculate*free chunks* of Remaining Memory by. 2. Create a script/tool that can iterate through each Diego Cell and do the following: a. Pull the `rep.CapacityRemainingMemory` metric for each cell. b. Divide the values received by 1000 to get the value in Gigabytes (if desired threshold is GB-based). c. Compare recorded values to your minimum capacity threshold, and count the number of cells that have equal or greater than the desired amount of *free chunk* space. 3. Determine a desired scaling threshold based on the minimum amount of *free chunks* that are acceptable in this deployment given historical trends. 4. Set an alert to indicate the need to scale cell memory capacity when the value falls below the desired threshold number.

	For visualization purposes: Looking at this metric (`rep.CapacityRemainingMemory`) as a minimum value per cell has more informational value than alerting value. It can be an interesting heatmap visualization, showing average variance and density over time.
Recommended alert thresholds	**Yellow warning**: Dynamic **Red critical**: Dynamic
Recommended response	1. Assign more resources to the cells or assign more cells. 2. Scale additional Diego cells using Ops Manager.

Remaining Memory Available — Overall Remaining Memory Available

rep.CapacityRemainingMemory (Alternative Use)	
Description	Remaining amount of memory in MiB available for this Diego cell to allocate to containers. **Use**: Can indicate low memory capacity overall in the platform. Low memory can prevent app scaling and new deployments. The overall sum of capacity can indicate that you need to scale the platform. Observing capacity consumption trends over time helps with capacity planning. **Origin**: Firehose **Type**: Gauge (Integer in bytes) **Frequency**: 60 s
Recommended measurement	Minimum over the last 5 minutes divided by 1024 (across all instances)
Recommended alert thresholds	**Yellow warning**: ≤ 64 GB **Red critical**: ≤ 32 GB
Recommended response	1. Assign more resources to the cells or assign more cells. 2. Scale additional Diego cells via Ops Manager.

Remaining Disk Available

rep.CapacityRemainingDisk	
Description	Remaining amount of disk in MiB available for this Diego cell to allocate to containers. **Use**: Low disk capacity can prevent app scaling and new deployments. Because Diego staging Tasks can fail without at least 4 GB free, the recommended red threshold is based on the minimum disk capacity across the deployment falling below 4 GB in the previous 5 minutes. It can also be meaningful to assess how many chunks of free disk space are above a given threshold, similar to `rep.CapacityRemainingMemory`. **Origin**: Firehose **Type**: Gauge (Integer in bytes) **Frequency**: 60 s
Recommended measurement	Minimum over the last 5 minutes divided by 1024 (across all instances)
Recommended alert thresholds	**Yellow warning**: ≤ 8 GB **Red critical**: ≤ 3.5 GB
Recommended response	1. Assign more resources to the cells or assign more cells. 2. Scale additional cells using Ops Manager.

Cell Rep Time to Sync

rep.RepBulkSyncDuration	
Description	Time in ns that the Diego Cell Rep took to sync the ActualLRPs that it claimed with its actual garden containers. **Use**: Sync times that are too high can indicate issues with the BBS. **Origin**: Firehose **Type**: Gauge (Float in ns) **Frequency**: 30 s
Recommended measurement	Maximum over the last 15 minutes divided by 1,000,000,000
Recommended alert thresholds	**Yellow warning**: ≥ 5 s **Red critical**: ≥ 10 s
Recommended response	1. Investigate BBS logs for faults and errors. 2. If a particular cell or cells appear problematic, pull logs for the cells and the BBS logs before contacting Pivotal Support.

Unhealthy Cells

rep.UnhealthyCell	
Description	The Diego cell periodically checks its health against the garden backend. For Diego cells, `0` means healthy, and `1` means unhealthy. **Use**: Set an alert for further investigation if multiple unhealthy Diego cells are detected in the given time window. If one cell is impacted, it does not participate in auctions, but end-user impact is usually low. If multiple cells are impacted, this can indicate a larger problem with Diego. Suggested alert threshold based on multiple unhealthy cells in the given time window. **Origin**: Firehose **Type**: Gauge (Float, 0-1) **Frequency**: 30 s
Recommended measurement	Maximum over the last 5 minutes
Recommended alert thresholds	**Yellow warning**: *N/A* **Red critical**: > 1
Recommended response	1. Investigate Diego cell servers for faults and errors. 2. If a particular cell or cells appear problematic, pull logs for that cell, as well as the BBS logs before contacting Pivotal Support.

Diego Locket Metrics

Active Locks

locket.ActiveLocks	
Description	Total count of how many locks the system components are holding. **Use**: If the ActiveLocks count is not equal to the expected value, there is likely a problem with Diego.

	Origin: Firehose **Type**: Gauge **Frequency**: 60 s
Recommended measurement	Maximum over the last 5 minutes
Recommended alert thresholds	**Yellow warning**: *N/A* **Red critical**: ≠ 4
Recommended response	1. Run `monit status` to inspect for failing processes. 2. If there are no failing processes, then review the logs for the components using the Locket service: BBS, Auctioneer, TPS Watcher, and Routing API. Look for indications that only one of each component is active at a time. 3. Focus triage on the BBS first: ◦ A healthy BBS shows obvious activity around starting or claiming LRPs. ◦ An unhealthy BBS leads to the Auctioneer showing minimal or no activity. The BBS sends work to the Auctioneer. ◦ Reference the BBS-level Locket metric Locks Held by BBS. A value of 0 indicates Locket issues at the BBS level. 4. If the BBS appears healthy, then check the Auctioneer to ensure it is processing auction payloads. ◦ Recent logs for Auctioneer should show all but one of its instances are currently waiting on locks, and the active Auctioneer should show a record of when it last attempted to execute work. This attempt should correspond to app development activity, such as `cf push`. ◦ Reference the Auctioneer-level Locket metric Locks Held by Auctioneer. A value of 0 indicates Locket issues at the Auctioneer level. 5. The TPS Watcher is primarily active when app instances crash. Therefore, if the TPS Watcher is suspected, review the most recent logs. 6. If you are unable to resolve on-going excessive active locks, pull logs from the Diego BBS and Auctioneer VMs, which includes the Locket service component logs, and contact Pivotal Support.

Locks Held by BBS

bbs.LockHeld	
Description	Whether a BBS instance holds the expected BBS lock (in Locket). 1 means the active BBS server holds the lock, and 0 means the lock was lost. **Use**: This metric is complimentary to Active Locks, and it offers a BBS-level version of the Locket metrics. Although it is emitted per BBS instance, only 1 active lock is held by BBS. Therefore, the expected value is 1. The metric may occasionally be 0 when the BBS instances are performing a leader transition, but a prolonged value of 0 indicates an issue with BBS. **Origin**: Firehose **Type**: Gauge **Frequency**: Periodically
Recommended measurement	Maximum over the last 5 minutes
Recommended alert thresholds	**Yellow warning**: *N/A* **Red critical**: ≠ 1
Recommended response	1. Run `monit status` on the Diego database VM to check for failing processes. 2. If there are no failing processes, then review the logs for BBS. ◦ A healthy BBS shows obvious activity around starting or claiming LRPs. ◦ An unhealthy BBS leads to the Auctioneer showing minimal or no activity. The BBS sends work to the Auctioneer. 3. If you are unable to resolve the issue, pull logs from the Diego BBS and Auctioneer VMs, which include the Locket service component logs, and contact Pivotal Support.

Locks Held by Auctioneer

auctioneer.LockHeld	
Description	Whether an Auctioneer instance holds the expected Auctioneer lock (in Locket). 1 means the active Auctioneer holds the lock, and 0 means the lock was lost. **Use**: This metric is complimentary to Active Locks, and it offers an Auctioneer-level version of the Locket metrics. Although it is emitted per Auctioneer instance, only 1 active lock is held by Auctioneer. Therefore, the expected value is 1. The metric may occasionally be 0 when the Auctioneer instances are performing a leader transition, but a prolonged value of 0 indicates an issue with Auctioneer. **Origin**: Firehose **Type**: Gauge **Frequency**: Periodically
Recommended measurement	Maximum over the last 5 minutes
Recommended alert thresholds	**Yellow warning**: *N/A* **Red critical**: ≠ 1
Recommended response	1. Run `monit status` on the Diego Database VM to check for failing processes. 2. If there are no failing processes, then review the logs for Auctioneer. ◦ Recent logs for Auctioneer should show all but one of its instances are currently waiting on locks, and the active Auctioneer should show a record of when it last attempted to execute work. This attempt should correspond to app development activity, such as `cf push`. 3. If you are unable to resolve the issue, pull logs from the Diego BBS and Auctioneer VMs, which includes the Locket service component logs, and contact Pivotal Support.

Active Presences

locket.ActivePresences	
Description	Total count of active presences. Presences are defined as the registration records that the cells maintain to advertise themselves to the platform. **Use**: If the Active Presences count is far from the expected, there might be a problem with Diego. The number of active presences varies according to the number of cells deployed. Therefore, during purposeful scale adjustments to PCF, this alerting threshold should be adjusted. Establish an initial threshold by observing the historical trends for the deployment over a brief period of time, Increase the threshold as more cells are deployed. During a rolling deploy, this metric shows variance during the BOSH lifecycle when cells are evacuated and restarted. Tolerable variance is within the bounds of the max inflight range, `Max Inflight Container Starts` established in Pivotal Application Service (PAS). **Origin**: Firehose **Type**: Gauge **Frequency**: 60 s
Recommended measurement	Maximum over the last 15 minutes
Recommended alert thresholds	**Yellow warning**: Dynamic **Red critical**: Dynamic
	1. Ensure that the variance is not the result of an active rolling deploy. Also ensure that the alert threshold is appropriate to the number of cells in the current deployment. 2. Run `monit status` to inspect for failing processes. 3. If there are no failing processes, then review the logs for the components using the Locket service: BBS, Auctioneer, TPS Watcher, and Routing API. 4. Focus triage on the BBS first:

Recommended response	◦ A healthy BBS shows obvious activity around starting or claiming LRPs. ◦ An unhealthy BBS leads to the Auctioneer showing minimal or no activity. The BBS sends work to the Auctioneer. 5. If the BBS appears healthy, then check the Auctioneer to ensure it is processing auction payloads. Recent logs for the active Auctioneer should show a record of when it last attempted to execute work. This attempt should correspond to app dev activity, such as a cf push. The TPS Watcher is primarily only active when application instances crash, so if the TPS Watcher is suspected, review the most recent logs. 6. If you are unable to resolve the problem, pull the logs from the Diego BBS and Auctioneer VMs, which include the Locket service component logs, and contact Pivotal Support.

Diego Route Emitter Metrics

Route Emitter Time to Sync

route_emitter.RouteEmitterSyncDuration	
Description	Time in ns that the active Route Emitter took to perform its synchronization pass. **Use**: Increases in this metric indicate that the Route Emitter may have trouble maintaining an accurate routing table to broadcast to the Gorouters. Tune alerting values to your deployment based on historical data and adjust based on observations over time. The suggested starting point is ≥ 5 for the yellow threshold and ≥ 10 for the critical threshold. Pivotal has observed on its Pivotal Web Services deployment that above 10 s, the BBS may be failing. **Origin**: Firehose **Type**: Gauge (Float in ns) **Frequency**: 60 s
Recommended measurement	Maximum, per job, over the last 15 minutes divided by 1,000,000,000
Recommended alert thresholds	**Yellow warning**: Dynamic **Red critical**: Dynamic
Recommended response	If all or many jobs showing as impacted, there is likely an issue with Diego. 1. Investigate the Route Emitter and Diego BBS logs for errors. 2. Verify that app routes are functional by making a request to an app, pushing an app and pinging it, or if applicable, checking that your smoke tests have passed. If one or a few jobs showing as impacted, there is likely a connectivity issue and the impacted job should be investigated further.

PAS MySQL KPIs

When PAS uses an internal MySQL database, as configured in the PAS tile **Settings** tab > **Databases** pane, the database cluster generates KPIs as described below.

MySQL Server Availability

	/mysql/available
Description	The MySQL Server is currently responding to requests, which indicates that the server is running. **Use:** This metric is especially useful in single-node mode, where cluster metrics are not relevant. If the server does not emit heartbeats, it is offline. **Origin:** Firehose

	/mysql/available
	Envelope Type: Gauge **Unit:** boolean **Frequency:** 30 s (default)
Recommended measurement	Average over the last 5 minutes
Recommended alert thresholds	**Yellow warning:** N/A **Red critical:** < 1
Recommended response	Run `mysql-diag` and check the MySQL Server logs for errors.

Galera Cluster Node Readiness

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

Galera Cluster Size

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

Galera Cluster Status

	/mysql/galera/wsrep_cluster_status
	Shows the primary status of the cluster component that the node is in. Values are: - Primary = 1

	/mysql/galera/wsrep_cluster_status
Description	- Non-primary = 0 - Disconnected = -1 See: https://mariadb.com/kb/en/mariadb/galera-cluster-status-variables/ **Use:** Any value other than “Primary” indicates that the node is part of a nonoperational component. This occurs in cases of multiple membership changes that result in a loss of quorum. **Origin:** Firehose **Envelope Type:** Gauge **Unit:** integer (see above) **Frequency:** 30 s (default)
Recommended measurement	Sum of each of the nodes, over the last 5 minutes
Recommended alert thresholds	**Yellow warning:** < 3 **Red critical:** < 1
Recommended response	- Check node status to ensure that they are all in working order and able to receive write-sets. - Run `mysql-diag` and check the MySQL Server logs for errors.

Connections per Second

	/mysql/net/connections
Description	Connections per second made to the server. **Use:** If the number of connections drastically changes or if apps are unable to connect, there might be a network or app issue. **Origin:** Firehose **Envelope Type:** Gauge **Unit:** count **Frequency:** 30 s (default)
Recommended measurement	(Average of all nodes / max connections), over last 1 minute
Recommended alert thresholds	**Yellow warning:** > 80% **Red critical:** > 90%
Recommended response	- Run `mysql-diag` and check the MySQL Server logs for errors. - When approaching 100% of max connections, Apps may be experiencing times when they cannot connect to the database. The connections per second for the cluster vary based on application instances and app utilization. If this threshold is met or exceeded for an extended period of time, monitor app usage to ensure everything is behaving as expected.

Query Rate

	/mysql/performance/questions
Description	The rate of statements execute by the server, shown as queries per second. **Use:** The cluster should always be processing some queries, if just as part of the internal automation. **Origin:** Firehose **Envelope Type:** Gauge **Unit:** count **Frequency:** 30 s (default)
Recommended measurement	Average over the last two minutes
Recommended alert thresholds	**Yellow warning:** 0 for 90 s **Red critical:** 0 for 120 s
Recommended response	If the rate is ever zero for an extended time, run `mysql-diag` and investigate the MySQL server logs to understand why query rate changed and determine appropriate action.

MySQL CPU Busy Time

	/mysql/performance/busy_time
Description	Percentage of CPU time spent by MySQL on user activity, executing user code, as opposed to kernel activity processing system calls. **Use:** This closely reflects the amount of server activity dedicated to app queries. **Origin:** Firehose **Envelope Type:** Gauge **Unit:** percentage **Frequency:** 30 s (default)
Recommended measurement	Average over last 2 minutes
Recommended alert thresholds	**Yellow warning:** > 80% **Red critical:** > 90%
Recommended response	- If this metric meets or exceeds the recommended thresholds for extended periods of time, run `SHOW PROCESSLIST` and identify which queries or apps are using so much CPU. Optionally redeploy the MySQL jobs using VMs with more CPU capacity. - Run `mysql-diag` and check the MySQL Server logs for errors.

Gorouter Metrics

Router File Descriptors

gorouter.file_descriptors	
Description	The number of file descriptors currently used by the Gorouter job. **Use**: Indicates an impending issue with the Gorouter. Without proper mitigation, it is possible for an unresponsive app to eventually exhaust available Gorouter file descriptors and cause route starvation for other apps running on PCF. Under heavy load, this unmitigated situation can also result in the Gorouter losing its connection to NATS and all routes being pruned. While a drop in `gorouter.total_routes` or an increase in `gorouter.ms_since_last_registry_update` helps to surface that the issue may already be occurring, alerting on `gorouter.file_descriptors` indicates that such an issue is impending. The Gorouter limits the number of file descriptors to 100,000 per job. Once the limit is met, the Gorouter is unable to establish any new connections. To reduce the risk of DDoS attacks, Pivotal recommends doing one or both of the following: • Within PAS, set **Max Connections Per Backend** to define how many requests can be routed to any particular app instance. This prevents a single app from using all Gorouter connections. The value specified should be determined by the operator based on the use cases for that foundation. For example, Pivotal sets the number of connections to 500 for Pivotal Web Services. • Add rate limiting at the load balancer level. **Origin**: Firehose **Type**: Gauge **Frequency**: 5 s
Recommended measurement	Maximum, per Gorouter job, over the last 5 minutes
Recommended alert thresholds	**Yellow warning**: 50,000 per job **Red critical**: 60,000 per job
Recommended response	1. Identify which app(s) are requesting excessive connections and resolve the impacting issues with these apps. 2. If the above recommended mitigation steps have not already been taken, do so.

	3. Consider adding more Gorouter VM resources to increase the number of available file descriptors.

Router Exhausted Connections

gorouter.backend_exhausted_conns	
Description	The lifetime number of requests that have been rejected by the Gorouter VM due to the `Max Connections Per Backend` limit being reached across all tried backends. The limit controls the number of concurrent TCP connections to any particular app instance and is configured within PAS. **Use**: Indicates that PCF is mitigating risk to other applications by self-protecting the platform against one or more unresponsive applications. Increases in this metric indicate the need to investigate and resolve issues with potentially unresponsive applications. A rapid rate of change upward is concerning and should be assessed further. **Origin**: Firehose **Type**: Counter (Integer) **Frequency**: 5 s
Recommended measurement	Maximum delta per minute, per Gorouter job, over a 5-minute window
Recommended alert thresholds	**Yellow warning**: Dynamic **Red critical**: Dynamic
Recommended response	1. If `gorouter.backend_exhausted_conns` spikes, first look to the Router Throughput metric `gorouter.total_requests` to determine if this measure is high or low in relation to normal bounds for this deployment. 2. If Router Throughput appears within normal bounds, it is likely that `gorouter.backend_exhausted_conns` is spiking due to an unresponsive application, possibly due to application code issues or underlying application dependency issues. To help determine the problematic application, look in access logs for repeated calls to one application. Then proceed to troubleshoot this application accordingly. 3. If Router Throughput also shows unusual spikes, the cause of the increase in `gorouter.backend_exhausted_conns` spikes is likely external to the platform. Unusual increases in load may be due to expected business events driving additional traffic to applications. Unexpected increases in load may indicate a DDoS attack risk.

Router Throughput

gorouter.total_requests	
Description	The lifetime number of requests completed by the Gorouter VM, emitted per Gorouter instance **Use**: The aggregation of these values across all Gorouters provide insight into the overall traffic flow of a deployment. Unusually high spikes, if not known to be associated with an expected increase in demand, could indicate a DDoS risk. For performance and capacity management, consider this metric a measure of router throughput per job, converting it to requests-per-second, by looking at the delta value of `gorouter.total_requests` and deriving back to 1s, or `gorouter.total_requests.delta)/5`, per Gorouter instance. This helps you see trends in the throughput rate that indicate a need to scale the Gorouter instances. Use the trends you observe to tune the threshold alerts for this metric. **Origin**: Firehose **Type**: Counter (Integer) **Frequency**: 5 s
Recommended measurement	Average over the last 5 minutes of the derived per second calculation
Recommended alert thresholds	**Yellow warning**: Dynamic **Red critical**: Dynamic

Recommended response	For optimizing the Gorouter, consider the requests-per-second derived metric in the context of router latency and Gorouter VM CPU utilization. From performance and load testing of the Gorouter, Pivotal has observed that at approximately 2500 requests per second, latency can begin to increase. To increase throughput and maintain low latency, scale the Gorouters either horizontally or vertically and watch that the system.cpu.user metric for the Gorouter stays in the suggested range of 60-70% CPU Utilization.

Router Handling Latency

gorouter.latency	
Description	The time in milliseconds that the Gorouter takes to handle requests to its app endpoints. This is the average round trip response time to an app, which includes router handling. **Use**: Indicates how Gorouter jobs in PCF are impacting overall app responsiveness. Latencies above 100 ms can indicate problems with the network, misbehaving apps, or the need to scale the Gorouter itself due to ongoing traffic congestion. An alert value on this metric should be tuned to the specifics of the deployment and its underlying network considerations; a suggested starting point is 100 ms. **Origin**: Firehose **Type**: Gauge (Float in ms) **Frequency**: Emitted per Gorouter request, emission should be constant on a running deployment
Recommended measurement	Average over the last 30 minutes
Recommended alert thresholds	**Yellow warning**: Dynamic **Red critical**: Dynamic
Recommended response	Extended periods of high latency can point to several factors. The Gorouter latency measure includes network and app latency impacts as well. 1. First inspect logs for network issues and indications of misbehaving apps. 2. If it appears that the Gorouter needs to scale due to ongoing traffic congestion, do not scale on the latency metric alone. You should also look at the CPU utilization of the Gorouter VMs and keep it within a maximum 60-70% range. 3. Resolve high utilization by scaling the Gorouter.

Time Since Last Route Register Received

gorouter.ms_since_last_registry_update	
Description	Time in milliseconds since the last route register was received, emitted per Gorouter instance **Use**: Indicates if routes are not being registered to apps correctly. **Origin**: Firehose **Type**: Gauge (Float in ms) **Frequency**: 30 s
Recommended measurement	Maximum over the last 5 minutes
Recommended alert thresholds	**Yellow warning**: *N/A* **Red critical**: > 30,000 This threshold is suitable for normal platform usage. It alerts if it has been at least 30 seconds since the Gorouter last received a message from an app.
Recommended response	1. Search the Gorouter and Route Emitter logs for connection issues to NATS. 2. Check the BOSH logs to see if the NATS, Gorouter, or Route Emitter VMs are failing. 3. Look more broadly at the health of all VMs, particularly Diego-related VMs.

	4. If problems persist, pull the Gorouter and Route Emitter logs and contact Pivotal Support to say there are consistently long delays in route registry.

Router Error: 502 Bad Gateway

gorouter.bad_gateways	
Description	The lifetime number of bad gateways, or 502 responses, from the Gorouter itself, emitted per Gorouter instance. The Gorouter emits a 502 bad gateway error when it has a route in the routing table and, in attempting to make a connection to the backend, finds that the backend does not exist. **Use**: Indicates that route tables might be stale. Stale routing tables suggest an issue in the route register management plane, which indicates that something has likely changed with the locations of the containers. Always investigate unexpected increases in this metric. **Origin**: Firehose **Type**: Count (Integer, Lifetime) **Frequency**: 5 s
Recommended measurement	Maximum delta per minute over a 5-minute window
Recommended alert thresholds	**Yellow warning**: Dynamic **Red critical**: Dynamic
Recommended response	1. Check the Gorouter and Route Emitter logs to see if they are experiencing issues when connecting to NATS. 2. Check the BOSH logs to see if the NATS, Gorouter, or Route Emitter VMs are failing. 3. Look broadly at the health of all VMs, particularly Diego-related VMs. 4. If problems persist, pull Gorouter and Route Emitter logs and contact Pivotal Support to say there has been an unusual increase in Gorouter bad gateway responses.

Router Error: Server Error

gorouter.responses.5xx	
Description	The lifetime number of requests completed by the Gorouter VM for HTTP status family 5xx, server errors, emitted per Gorouter instance. **Use**: A repeatedly crashing app is often the cause of a big increase in 5xx responses. However, response issues from apps can also cause an increase in 5xx responses. Always investigate an unexpected increase in this metric. **Origin**: Firehose **Type**: Counter (Integer) **Frequency**: 5 s
Recommended measurement	Maximum delta per minute over a 5-minute window
Recommended alert thresholds	**Yellow warning**: Dynamic **Red critical**: Dynamic
Recommended response	1. Look for out-of-memory errors and other app-level errors. 2. As a temporary measure, ensure that the troublesome app is scaled to more than one instance.

Number of Gorouter Routes Registered

gorouter.total_routes	
Description	The current total number of routes registered with the Gorouter, emitted per Gorouter instance **Use**: The aggregation of these values across all Gorouters indicates uptake and gives a picture of the overall growth of the environment for capacity planning. Pivotal also recommends alerting on this metric if the number of routes falls outside of the normal range for your deployment. Dramatic decreases in this metric volume may indicate a problem with the route registration process, such as an app outage, or that something in the route register management plane has failed. If visualizing these metrics on a dashboard, `gorouter.total_routes` can be helpful for visualizing dramatic drops. However, for alerting purposes, the `gorouter.ms_since_last_registry_update` metric is more valuable for quicker identification of Gorouter issues. Alerting thresholds for `gorouter.total_routes` should focus on dramatic increases or decreases out of expected range. **Origin**: Firehose **Type**: Gauge (Float) **Frequency**: 30 s
Recommended measurement	5-minute average of the per second delta
Recommended alert thresholds	**Yellow warning**: Dynamic **Red critical**: Dynamic
Recommended response	1. For capacity needs, scale up or down the Gorouter VMs as necessary. 2. For significant drops in current total routes, see the `gorouter.ms_since_last_registry_update` metric value for additional context. 3. Check the Gorouter and Route Emitter logs to see if they are experiencing issues when connecting to NATS. 4. Check the BOSH logs to see if the NATS, Gorouter, or Route Emitter VMs are failing. 5. Look broadly at the health of all VMs, particularly Diego-related VMs. 6. If problems persist, pull the Gorouter and Route Emitter logs and contact Pivotal Support.

Number of Route Registration Messages Sent and Received

gorouter.registry_message.route-emitter route_emitter.MessagesEmitted	
Description	This KPI is based on the following metrics: • `route_emitter.MessagesEmitted` reports the lifetime number of route registration messages sent by the Route Emitter component. The metric is emitted for each Route Emitter. • `gorouter.registry_message.route-emitter` reports the lifetime number of route registration messages received by the Gorouter. The metric is emitted for each Gorouter instance. Dynamic configuration that enables the Gorouter to route HTTP requests to apps is published by the Route Emitter component colocated on each Diego cell to the NATS clustered message bus. All router instances subscribed to this message bus receive the same configuration. (Router instances within an isolation segment receive configuration only for cells in the same isolation segment.) As Gorouters prune app instances from the route when a TTL expires, each Route Emitter periodically publishes the routing configuration for the app instances on the same cell. Therefore, the aggregate number of route registration messages published by all the Route Emitters should be equal to the number of messages received by each Gorouter instance. **Use**: A difference in the rate of change of these metrics is an indication of an issue in the control plane responsible for updating the routers with changes to the routing table. Pivotal recommends alerting when the number of messages received per second for a given router instance falls

	below the sum of messages emitted per second across all Route Emitters. If visualizing these metrics on a dashboard, look for increases in the difference between the rate of messages received and sent. If the number of messages received by a Gorouter instance drops below the sum of messages sent by the Route Emitters, this is an indication of a problem in the control plane. **Origin**: Firehose **Type**: Counter **Frequency:** With each event
Recommended measurement	Difference of 5-minute average of the per second deltas for `gorouter.registry_message.route-emitter` and sum of `route_emitter.MessagesEmitted` for all Route Emitters
Recommended alert thresholds	**Yellow warning**: Dynamic **Red critical**: Dynamic
Recommended response	1. Check the Gorouter and Route Emitter logs to see if they are experiencing issues when connecting to NATS. 2. Check the BOSH logs to see if the NATS, Gorouter, or Route Emitter VMs are failing. 3. Look broadly at the health of all VMs, particularly Diego-related VMs. 4. If problems persist, pull the Gorouter and Route Emitter logs and contact Pivotal Support.

Firehose Metrics

Firehose Throughput

DopplerServer.listeners.totalReceivedMessageCount + loggregator.doppler.ingress	
Description	The total number of messages received across all Doppler listeners: UDP, TCP, TLS, and GRPC. **Use**: Provides insight into how much traffic the logging system handles. This metric is an indicator of logging consistency. **Origin**: Firehose **Type**: Counter (Integer) **Frequency:** 5 s
Recommended measurement	Maximum delta per minute over a 5-minute window
Recommended alert thresholds	**Yellow warning**: Dynamic **Red critical**: Dynamic
Recommended response	Scale up the Firehose log receiver and Dopplers on consistent upward trends. Pivotal recommends that you do not scale down these components on flat or downward delta trends because unexpected spikes in throughput can cause log loss if not scaled appropriately.

Firehose Dropped Messages

DopplerServer.doppler.shedEnvelopes + loggregator.doppler.dropped	
Description	The lifetime total number of messages intentionally dropped by Doppler due to back pressure. **Use**: Indicates logging consistency. Set an alert to indicate if too much traffic is coming into the Dopplers or if the Firehose consumers are not keeping pace. Both issues result in dropped messages. **Origin**: Firehose **Type**: Counter (Integer) **Frequency:** 5 s
	Maximum delta per minute over a 5-minute window

Recommended measurement **Recommended alert thresholds**	**Yellow warning:** ≥ 5 **Red critical:** ≥ 10
Recommended response	Scale up the Firehose log receiver and Dopplers.

System (BOSH) Metrics

VM Health

system.healthy	
Description	`1` means the system is healthy, and `0` means the system is not healthy. **Use:** This is the most important BOSH metric to monitor. It indicates if the VM emitting the metric is healthy. Review this metric for all VMs to estimate the overall health of the system. Multiple unhealthy VMs signals problems with the underlying IAAS layer. **Origin:** Firehose **Type:** Gauge (Float, 0-1) **Frequency:** 60 s
Recommended measurement	Average over the last 5 minutes
Recommended alert thresholds	**Yellow warning:** *N/A* **Red critical:** < 1
Recommended response	Investigate CF logs for the unhealthy component(s).

VM Memory Used

system.mem.percent	
Description	System Memory — Percentage of memory used on the VM **Use:** Set an alert and investigate if the free RAM is low over an extended period. **Origin:** Firehose **Type:** Gauge (%) **Frequency:** 60 s
Recommended measurement	Average over the last 10 minutes
Recommended alert thresholds	**Yellow warning:** ≥ 80% **Red critical:** ≥ 90%
Recommended response	The response depends on the job the metric is associated with. If appropriate, scale affected jobs out and monitor for improvement.

VM Disk Used

system.disk.system.percent	
Description	System disk — Percentage of the system disk used on the VM **Use:** Set an alert to indicate when the system disk is almost full. **Origin:** Firehose **Type:** Gauge (%)

	Frequency: 60 s
Recommended measurement	Average over the last 30 minutes
Recommended alert thresholds	**Yellow warning**: ≥ 80% **Red critical**: ≥ 90%
Recommended response	Investigate what is filling the jobs system partition. This partition should not typically fill because BOSH deploys jobs to use ephemeral and persistent disks.

VM Ephemeral Disk Used

system.disk.ephemeral.percent	
Description	Ephemeral disk — Percentage of the ephemeral disk used on the VM **Use**: Set an alert and investigate if the ephemeral disk usage is too high for a job over an extended period. **Origin**: Firehose **Type**: Gauge (%) **Frequency**: 60 s
Recommended measurement	Average over the last 30 minutes
Recommended alert thresholds	**Yellow warning**: ≥ 80% **Red critical**: ≥ 90%
Recommended response	1. Run `bosh vms --details` to view jobs on affected deployments. 2. Determine cause of the data consumption, and, if appropriate, increase disk space or scale out the affected jobs.

VM Persistent Disk Used

system.disk.persistent.percent	
Description	Persistent disk — Percentage of persistent disk used on the VM **Use**: Set an alert and investigate further if the persistent disk usage for a job is too high over an extended period. **Origin**: Firehose **Type**: Gauge (%) **Frequency**: 60 s
Recommended measurement	Average over the last 30 minutes
Recommended alert thresholds	**Yellow warning**: ≥ 80% **Red critical**: ≥ 90%
Recommended response	1. Run `bosh vms --details` to view jobs on affected deployments. 2. Determine cause of the data consumption, and, if appropriate, increase disk space or scale out affected jobs.

VM CPU Utilization

system.cpu.user	
	CPU utilization — The percentage of CPU spent in user processes **Use**: Set an alert and investigate further if the CPU utilization is too high for a job.

Description	For monitoring Gorouter performance, CPU utilization of the Gorouter VM is the recommended key capacity scaling indicator. For more information, see Gorouter Latency and Throughput. **Origin**: Firehose **Type**: Gauge (%) **Frequency**: 60 s
Recommended measurement	Average over the last 5 minutes
Recommended alert thresholds	**Yellow warning**: ≥ 85% **Red critical**: ≥ 95%
Recommended response	1. Investigate the cause of the spike. 2. If the cause is a normal workload increase, then scale up the affected jobs.

Key Capacity Scaling Indicators

This topic describes key capacity scaling indicators that operators monitor to determine when they need to scale their Pivotal Cloud Foundry (PCF) deployments.

Pivotal provides these indicators to operators as general guidance for capacity scaling. Each indicator is based on platform metrics from different components. This guidance is applicable to most PCF v1.12 deployments. Pivotal recommends that operators fine-tune the suggested alert thresholds by observing historical trends for their deployments.

Diego Cell Capacity Scaling Indicators

There are three key capacity scaling indicators recommended for Diego cell:

- Diego Cell Memory Capacity is a measure of the percentage of remaining memory capacity
- Diego Cell Disk Capacity is a measure of the percentage of remaining disk capacity
- Diego Cell Container Capacity is a measure of the percentage of remaining container capacity

Diego Cell Memory Capacity

rep.CapacityRemainingMemory / rep.CapacityTotalMemory	
Description	Percentage of remaining memory capacity for a given cell. Monitor this derived metric across all cells in a deployment. The metric `rep.CapacityRemainingMemory` indicates the remaining amount in MiB of memory available for this cell to allocate to containers. The metric `rep.CapacityTotalMemory` indicates the total amount in MiB of memory available for this cell to allocate to containers.
Purpose	A best practice deployment of Cloud Foundry includes three availability zones (AZs). For these types of deployments, Pivotal recommends that you have enough capacity to suffer failure of an entire AZ. The *Recommended threshold* assumes a three-AZ configuration. Adjust the threshold percentage if you have more or fewer AZs.
Recommended thresholds	< avg(35%)
How to scale	Scale up your Diego Cells
Additional details	**Origin**: Firehose **Type**: Gauge (%) **Frequency**: Emitted every 60 s **Applies to**: cf:diego_cells

Diego Cell Disk Capacity

rep.CapacityRemainingDisk / rep.CapacityTotalDisk	
Description	Percentage of remaining disk capacity for a given cell. Monitor this derived metric across all cells in a deployment. The metric `rep.CapacityRemainingDisk` indicates the remaining amount in MiB of disk available for this cell to allocate to containers. The metric `rep.CapacityTotalDisk` indicates the total amount in MiB of disk available for this cell to allocate to containers.
Purpose	A best practice deployment of Cloud Foundry includes three availability zones (AZs). For these types of deployments, Pivotal recommends that you have enough capacity to suffer failure of an entire AZ. The *Recommended threshold* assumes a three-AZ configuration. Adjust the threshold percentage if you have

	more or fewer AZs.
Recommended thresholds	< avg(35%)
How to scale	Scale up your Diego Cells
Additional details	**Origin**: Firehose **Type**: Gauge (%) **Frequency**: Emitted every 60 s **Applies to**: cf:diego_cells

Diego Cell Container Capacity

rep.CapacityRemainingContainers / rep.CapacityTotalContainers	
Description	Percentage of remaining container capacity for a given cell. Monitor this derived metric across all cells in a deployment. The metric `rep.CapacityRemainingContainers` indicates the remaining number of containers this cell can host. The metric by `rep.CapacityTotalContainer` indicates the total number of containers this cell can host.
Purpose	A best practice deployment of Cloud Foundry includes three availability zones (AZs). For these types of deployments, Pivotal recommends that you have enough capacity to suffer failure of an entire AZ. The *Recommended threshold* assumes a three-AZ configuration. Adjust the threshold percentage if you have more or fewer AZs.
Recommended thresholds	< avg(35%)
How to scale	Scale up your Diego Cells
Additional details	**Origin**: Firehose **Type**: Gauge (%) **Frequency**: Emitted every 60 s **Applies to**: cf:diego_cells

Firehose Performance Scaling Indicator

There is one key capacity scaling indicator recommended for Firehose performance.

Firehose Loss Rate

(DopplerServer.doppler.shedEnvelopes + loggregator.doppler.dropped) / (DopplerServer.listeners.totalReceivedMessageCount + loggregator.doppler.ingress)	
Description	This derived value represents the Firehose loss rate, or the total messages dropped as a percentage of the total message throughput. Total messages include the combined stream of logs from all apps and the metrics data from Cloud Foundry components.
Purpose	Excessive dropped messages can indicate the Dopplers and/or Traffic Controllers are not processing messages fast enough. The recommended scaling indicator is to look at the total dropped as a percentage of the total throughput and scale if the derived loss rate value grows greater than `0.1`.
Recommended thresholds	**Scale indicator**: ≥ 0.1 If alerting: **Yellow warning**: ≥ 0.05 **Red critical**: ≥ 0.1
How to scale	Scale up the Firehose Traffic Controllers and Dopplers.
Additional details	**Origin**: Firehose **Type**: Gauge (float) **Frequency**: Base metrics are emitted every 5 s **Applies to**: cf:doppler

Pivotal

CF Syslog Drain Performance Scaling Indicators

There are three key capacity scaling indicators recommended for CF Syslog Drain performance.

Note: These CF Syslog Drain scaling indicators are only relevant if your deployment contains apps using the CF syslog drain binding feature.

Adapter Loss Rate

cf-syslog-drain.adapter.dropped / cf-syslog-drain.adapter.ingress	
Description	The loss rate of the Syslog Adapters, that is, the total messages dropped as a percentage of the total traffic coming through the Syslog Adapters. Total messages include only logs for bound applications. This loss rate is specific to the Syslog Adapters and does not impact the Firehose loss rate. For example, you can suffer lossiness in syslog while not suffering any lossiness in the Firehose.
Purpose	Indicates that the syslog drains are not keeping up with the number of logs that a syslog-drain-bound app is producing. This likely means that the syslog-drain consumer is failing to keep up with the incoming log volume. The recommended scaling indicator is to look at the maximum per minute loss rate over a 5-minute window and scale if the derived loss rate value grows greater than `0.1`.
Recommended thresholds	**Scale indicator**: ≥ 0.1 If alerting: **Yellow warning**: ≥ 0.01 **Red critical**: ≥ 0.1
How to scale	Performance test your syslog server, review the logs of the syslog consuming system for intake and other performance issues that indicate a need to scale the consuming system.
Additional details	**Origin**: Firehose **Type**: Counter (Integer) **Frequency**: Emitted every 60 s **Applies to**: cf:cf-syslog

Reverse Log Proxy Loss Rate

loggregator.rlp.dropped / loggregator.rlp.ingress	
Description	The loss rate of the reverse log proxies (RLP), that is, the total messages dropped as a percentage of the total traffic coming through the reverse log proxy. Total messages include only logs for bound applications. This loss rate is specific to the RLP and does not impact the Firehose loss rate. For example, you can suffer lossiness in syslog while not suffering any lossiness in the Firehose.
Purpose	Excessive dropped messages can indicate that the RLP is overloaded and that the Traffic Controllers need to be scaled. The recommended scaling indicator is to look at the maximum per minute loss rate over a 5-minute window and scale if the derived loss rate value grows greater than `0.1`.
Recommended thresholds	**Scale indicator**: ≥ 0.1 If alerting: **Yellow warning**: ≥ 0.01 **Red critical**: ≥ 0.1
How to scale	Scale up the number of traffic controller instances to further balance log load.
Additional details	**Origin**: Firehose **Type**: Counter (Integer) **Frequency**: Emitted every 60 s **Applies to**: cf:cf-syslog

CF Syslog Drain Bindings Count

cf-syslog-drain.scheduler.drains	

Description	The number of CF syslog drain bindings.
Purpose	Each Syslog Adapter can handle approximately 500 drain bindings. The recommended initial configuration is a minimum of two Syslog Adapters (to handle approximately 1000 drain bindings). A new Adapter instance should be added for each 500 additional drain bindings. Therefore, the recommended initial scaling indicator is 900 (as a maximum value over a 1-hr window). This indicates the need to scale up to three Adapters from the initial two-Adapter configuration.
Recommended thresholds	**Scale indicator**: ≥ 900 Consider this threshold to be dynamic. Adjust the threshold to the PCF deployment as adoption of CF Syslog Drain increases or decreases.
How to scale	Increase the number of Syslog Adapter VMs in the **Resource Config** pane of the Pivotal Application Service (PAS) tile.
Additional details	**Origin**: Firehose **Type**: Gauge (float) **Frequency**: Emitted every 60 s **Applies to**: cf:cf-syslog

Router Performance Scaling Indicator

There is one key capacity scaling indicator recommended for Router performance.

Router VM CPU Utilization

system.cpu.user of the Gorouter VM(s)	
Description	CPU utilization of the Gorouter VM(s)
Purpose	High CPU utilization of the Gorouter VMs can increase latency and cause throughput, or requests per/second, to level-off. Pivotal recommends keeping the CPU utilization within a maximum range of 60-70% for best Gorouter performance. If you want to increase throughput capabilities while also keeping latency low, Pivotal recommends scaling the Gorouter while continuing to ensure that CPU utilization does not exceed the maximum recommended range.
Recommended thresholds	**Scale indicator**: ≥ 60% If alerting: **Yellow warning**: ≥ 60% **Red critical**: ≥ 70%
How to scale	Resolve high utilization by scaling the Gorouters horizontally or vertically by editing the **Router** VM in the **Resource Config** pane of the PAS tile.
Additional details	**Origin**: Firehose **Type**: Gauge (float) **Frequency**: Emitted every 60 s **Applies to**: cf:router

NFS/WebDAV Backed Blobstore

There is one key capacity scaling indicator for external S3 external storage.

Note: This metric is only relevant if your deployment does not use an external S3 repository for external storage with no capacity constraints.

system.disk.persistent.percent of NFS server VM(s)	
Description	*If applicable*: Monitor the percentage of persistent disk used on the VM for the NFS Server job.
Purpose	If you do not use an external S3 repository for external storage with no capacity constraints, you must monitor the PCF object store to push new app and buildpacks. If you use an internal NFS/WebDAV backed blobstore, consider scaling the persistent disk when it reaches 75% capacity.

Recommended thresholds	≥ 75%
How to scale	Give your NFS Server additional persistent disk resources.
Additional details	**Origin**: Firehose **Type**: Gauge (%) **Applies to**: cf:nfs_server

Configuring a Monitoring System

This topic describes how to set up Pivotal Cloud Foundry (PCF) with third-party monitoring platforms to continuously monitor system metrics and trigger health alerts.

To perform a manual, one-time check of current PCF system status from Ops Manager, see Monitoring Virtual Machines in Pivotal Cloud Foundry.

Pivotal recommends that operators experiment with different combinations of metrics and alerts appropriate to their specific requirements. As an example, the Datadog Config repository shows how the Pivotal Cloud Ops team monitors the health of its Cloud Foundry deployments using a customized Datadog dashboard.

> **Note**: Pivotal does not support any monitoring platforms.

Overview

As a prerequisite to PCF monitoring, you need an account with a monitoring platform such as SignalFx, Datadog, OpenTDSB, New Relic, vRealize Operations (vROPS), AppDynamics, Dynatrace, or another tool.

To set up PCF monitoring, you then configure PCF and your monitoring platform as follows:

- In PCF:
  - Install a nozzle that extracts BOSH and CF metrics from the Firehose and sends them to the monitoring platform.
  - (Optional) Deploy smoke tests or other apps that generate custom metrics. Pivotal recommends custom metrics for production environments.
- In your monitoring platform:
  - Customize a dashboard that lets you check and diagnose system health.
  - Create alerts that generate communications regarding attention-worthy conditions.

BOSH Health Monitor and CF Component Metrics

You can configure PCF to direct metrics from all Pivotal Application Service (PAS) component VMs, including system components and hosts, to a monitoring platform. To do this, you configure component logs and metrics to stream from the Loggregator Firehose endpoint and install a nozzle that filters out the logs and directs the metrics to the monitoring platform.

The Loggregator Firehose metrics come from two sources:

- **The BOSH Health Monitor**: The BOSH Health Monitor receives health monitoring metrics from BOSH Agents for all VMs in a deployment.

  > **Note:** BOSH-reported system metrics are now available in the Loggregator Firehose by default. If your monitoring tool is set up to consume metrics from both the Loggregator Firehose and the PCF JMX Bridge tile, you may receive duplicate data. To prevent this, stop using PCF JMX Bridge to consume BOSH-reported system metrics outside of the Firehose. For more guidance, see BOSH System Metrics Available in Loggregator Firehose.

- **Cloud Foundry components**: Cloud Foundry component VMs for executive control, hosting, routing, traffic control, authentication, and other internal functions generate metrics.

> **Note:** PCF components specific to your IaaS also generate key metrics for health monitoring.

See the following topics for lists of high-signal-value metrics and capacity scaling indicators in a PCF deployment:

- Key Performance Indicators
- Key Capacity Scaling Indicators

Metrics Path from Component to Firehose

PCF component metrics originate from the Metron agents on their source components, then travel through Dopplers to the Traffic Controller.

The Traffic Controller aggregates both metrics and log messages system-wide from all Dopplers, and emits them from its Firehose endpoint.

Smoke Tests and Custom System Metrics

PCF includes smoke tests, which are functional unit and integration tests on all major system components. By default, whenever an operator upgrades to a new version of PAS, these smoke tests run as a post-deploy errand.

Production systems typically also have an app that runs smoke tests periodically, for example every five minutes, and generate "pass/fail" metrics from the results. For example smoke tests, see the Pivotal Cloud Ops CF Smoke Tests repository.

Operators can also generate other custom system metrics based on multi-component tests. An example is average outbound latency between components.

PCF Monitoring Setup

Perform the following steps to set up PCF monitoring:

1. Install a nozzle that extracts BOSH and CF metrics from the Loggregator Firehose and sends them to the monitoring platform.
2. Install a custom app to generate smoke test or other custom system metrics.
3. Customize your monitoring platform dashboard and alerts.

Install a Nozzle

To monitor BOSH and CF component metrics, you install a nozzle that directs the metrics from the Firehose to your monitoring platform. The nozzle process takes the Firehose output, ignores the logs, and sends the metrics.

You can see an example nozzle for sending metrics to Datadog in the datadog-firehose-nozzle GitHub repository. You configure the Datadog account credentials, API location, and other fields and options in the `config/datadog-firehose-nozzle.json` file.

Deploy a Custom System Metrics App

For production systems, Pivotal recommends deploying an app that runs regular smoke tests and other custom tests and generates metrics from the results.

A custom system metrics app sends metrics to the monitoring platform directly, so you must configure it with your platform endpoint and account information. The app does not run a Metron agent, and the Firehose does not carry custom system metrics.

The app can run in own Docker container, on a Concourse VM, or elsewhere.

See the Pivotal Cloud Ops CF Smoke Tests repository for more information and examples of smoke test and custom system metrics apps.

See the Metrics topic in the Concourse documentation for how to set up Concourse to generate custom system metrics.

Configure your Monitoring Platform

Monitoring platforms support two types of monitoring:

- A *dashboard* for active monitoring when you are at a keyboard and screen
- Automated *alerts* for when your attention is elsewhere

Some monitoring solutions offer both in one package. Others require putting the two pieces together.

See the Datadog Config repository for an example of how to configure a dashboard and alerts for Cloud Foundry in Datadog.

Determine Warning and Critical Thresholds

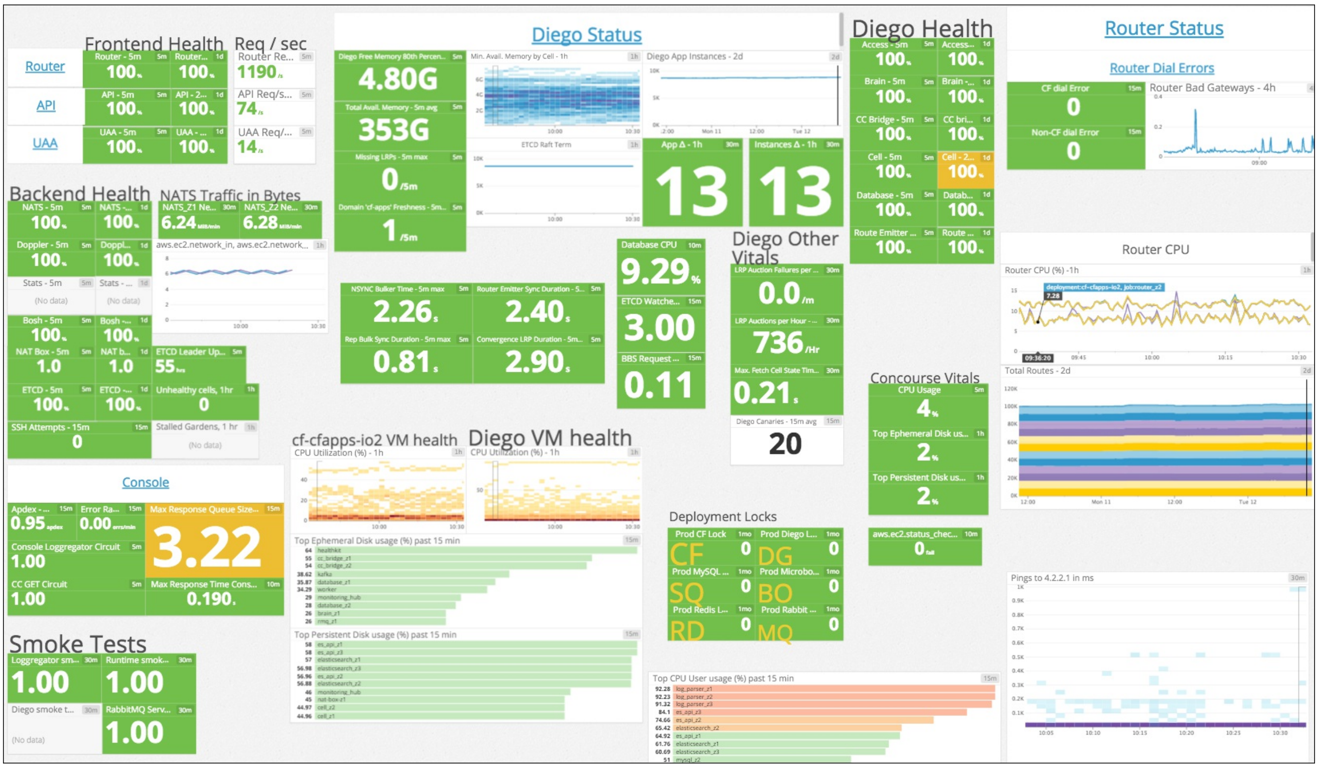
To properly configure your monitoring dashboard and alerts, you need to establish what thresholds should drive alerting and red/yellow/green dashboard behavior. PCF users can refer to these recommended [Key Performance Indicators] and [Key Capacity Scaling Indicators].

Some key metrics of operational value have more fixed thresholds, and recommended threshold numbers for them are generally the same values across different foundations and use cases. These metrics tend to revolve around the health and performance of key components that can impact the performance of the entire system.

Other key metrics of operational value are more dynamic in nature. This means that you must establish a baseline and yellow/red thresholds suitable for your system and its use cases. Initially, baselines can be established by watching values of key metrics over time and noting what seems to be a good starting threshold level between acceptable and unacceptable system performance and health. The initial baselines should continue to be refined as you determine the appropriate balance between early detection and reducing unnecessary alert fatigue. These types of more dynamic measures should be revisited on occasion to ensure they are still appropriate to the current system configuration and its usage patterns.

Customize Your Dashboard

You customize a dashboard by defining elements on the screen that show values derived from one or more metrics. These dashboard elements typically use simple formulas, such as averaging metric values over the past 60 seconds or summing them up over related instances. They are also often normalized to display with 3 or fewer digits for easy reading and color-coded red, yellow, or green to indicate health conditions.



In Datadog, for example, you can define a `screen_template` query that watches the `auctioneer.AuctioneerLRPAuctionsFailed` metric and displays its current average over the past minute.

Create Alerts

You create alerts by defining boolean conditions based on operations over one or more metrics, and an action that the platform takes when an alert triggers. The booleans typically check whether metric values exceed or fall below thresholds, or compare metric values against each other.

[In Datadog], for example, you can [define an `alert_template` condition] that triggers when the `auctioneer.AuctioneerLRPAuctionsFailed` metric indicates an average of more than one failed auction per minute for the past 15 minutes:

```
{
  "query": "min(last_15m):per_minute(avg:datadog.nozzle.auctioneer.AuctioneerLRPAuctionsFailed{deployment:<%= metron_agent_diego_deployment %>
  "message": "##Description:\nDiego internal metrics \n\n## Escalation Path:\nDiego \n\n## Possible Causes:\nThose alerts were a pretty strong
  "name": "<%= environment %> Diego: LRP Auction Failure per min is too high",
  "no_data_timeframe": 30,
```

```
    "notify_no_data": false
}
```

Actions that an alert triggers can include sending a pager or SMS message, sending an email, generating a support ticket, or passing the alert to a alerting system such as PagerDuty.

Monitoring Platforms

Some monitoring solutions offer both alerts and a dashboard in one package, while others require separate packages for alerts and dashboard.

Popular monitoring platforms among PCF customers include the following:

- AppDynamics
- Datadog
- Dynatrace
- New Relic
- OpenTSDB alerts and Grafana dashboard
- vRealize Operations (vROPS)

Backing Up and Restoring Pivotal Cloud Foundry

Page last updated:

Consider the following when backing up data in your Pivotal Cloud Foundry (PCF) deployment:

- If your deployment uses external databases, such as Amazon Web Services (AWS) RDS, you must back up your data according to the instructions provided by your database vendor.
- If your PCF deployment uses internal databases, follow the backup and restore instructions for the MySQL server included in the BBR documentation.

Backup and Restore with BBR

BOSH Backup and Restore (BBR) is a command-line tool for backing up and restoring BOSH deployments.

Note: The previously documented manual backup and restore procedures no longer work in PCF 1.12 because Ops Manager export and import no longer includes releases. You must back up and restore your BOSH Director using BBR to ensure releases are backed up and restored.

Note: You can only use BBR to back up PCF v1.11 and later.

To perform a backup of your PCF deployment with BBR, see Backing Up Pivotal Cloud Foundry with BBR.

To restore your PCF deployment with BBR, see Restoring Pivotal Cloud Foundry from Backup with BBR.

To use BBR to back up and restore your PCF deployment, you must first set up a jumpbox to run BBR from. See Setting Up Your Jumpbox for BBR.

To troubleshoot problems with BBR, see Troubleshooting BBR.

Disaster Recovery in Pivotal Cloud Foundry

This document provides an overview of the options and considerations for disaster recovery in Pivotal Cloud Foundry (PCF).

Operators have a range of approaches for ensuring they can recover Pivotal Cloud Foundry, apps, and data in case of a disaster. The approaches fall into the following two categories:

- Using data from a backup to restore the data in the PCF Deployment. See Back up and Restore Using BOSH Backup and Restore (BBR) for more information.
- Recreating the data in PCF by automating the creation of state in PCF. See Disaster Recovery by Recreating the Deployment for more information.

Back up and Restore using BOSH Backup and Restore (BBR)

What is BBR?

BOSH Backup and Restore (BBR) is a CLI for orchestrating backing up and restoring BOSH deployments and BOSH Directors. BBR triggers the backup or restore process on the deployment or Director, and transfers the backup artifact to and from the deployment or Director.

Use BOSH Backup and Restore to reliably create backups of core PCF components and their data. These core components include CredHub, UAA, BOSH Director, and PAS.

Each component includes its own backup scripts. This decentralized structure helps keep scripts synchronized with the components. At the same time, *locking* features ensure data integrity and consistent, distributed backups across your deployment.

How Does BBR Work?

1. An operator trigger a backup or a restore for a BOSH deployment or BOSH Director using the BBR binary on a jumpbox or in Concourse.

2. BOSH Backup and Restore examines the jobs in the deployment or Director for backup or restore scripts, then triggers those scripts in the prescribed order.

3. The artifacts are then transferred to or from the jumpbox or Concourse worker.

4. The operator must transfer artifacts to or from external storage.

Backing up PCF

Backing up PCF requires backing up the following components:

- Ops Manager settings
- BOSH Director, including CredHub and UAA
- Pivotal Application Service
- Data services

For more information, see Backing up Pivotal Cloud Foundry with BBR. With these backup artifacts, operators can recreate PCF exactly as it was when the backup was taken.

Restoring PCF

The restore process involves creating a new PCF deployment starting with the Ops Manager VM. For more information, see Restoring Pivotal Cloud Foundry from Backup with BBR.

The time required to restore the data is proportionate to the size of the data because the restore process includes copying data. For example, restoring a 1 TB blobstore takes one thousand times as long as restoring a 1 GB blobstore.

Benefits

Unlike other backup solutions, using BBR to back up PCF enables the following:

- **Completeness**: BBR supports backing up BOSH, including releases, CredHub, UAA, and service instances created with an on-demand service broker. With PCF v1.12, Ops Manager export no longer includes releases.
- **Consistency**: BBR provides referential integrity between the database and the blobstore because a lock is held while both the database and blobstore are backed up.
- **Correctness**: Using the BBR restore flow addresses C2C and routing issues that can occur during restore.

API Downtime During Backups

Apps are not affected during backups, but certain APIs are unavailable. The downtime occurs only while the backup is being taken, not while the backup is being copied to the jumpbox.

In a consistent backup, the blobs in the blobstore match the blobs in the Cloud Controller Database. To take a consistent backup, changes to the data are prevented during the backup. This means that the CF API, Routing API, Usage Service, Autoscaler, Notification Service, Network Policy Server, and CredHub are unavailable while the backup is being taken. UAA is in read-only mode during the backup.

Backup Timings

The table below gives an indication of the downtime that you can expect. Actual downtime varies based on hardware and PCF configuration.

These example timings were taken on GCP. The components scaled were CAPI workers, CredHub, UAA, Router, CAPI API, and Networking to simulate a “normal” and “very large” PCF deployment. API downtime is the sum of the time spent in lock, backup, and unlock phases.

Backup Timings with Scaled Components

	Backup phase	*Components scaled to 3*	*Components scaled to 6*
API unavailable	lock	9 minutes	21 minutes
	backup	<1 minute	<1 minute
	unlock	9 minutes	20 minutes
	drain and checksum	80 minutes	80 minutes

Blobstore backup and restore

Blobstores can be very large. To minimize downtime, only metadata about the blobs is taken during the back up. For example, in the case of internal blobstores (Webdav/NFS), a list of hardlinks to the blobs is taken. After API access is restored, copies of the blobs are made.

Products and Hardware Not Supported

The follow components and products do not yet support BBR:

- **Data services**: BBR is not yet supported in Pivotal’s flagship data services (MySQL, RabbitMQ, Redis, PCC). In the meantime, operators should use the automatic backups feature of each tile, available within Ops Manager.
- **External blobstores**: BBR does not yet back up external blobstores but can be used to back up the rest of PAS. Pivotal recommends that operators copy the blobstore using IaaS tooling, in conjunction with backing up PAS with BBR.
- **External databases**: BBR does not yet support PAS configured with an external database. Pivotal recommends that operators copy the database using IaaS tooling.

To address the limitations noted above, follow the guidlines below when using BBR to back up PCF when PAS configured with an external blobstore or external database:

- With PAS configured with an internal database and an external blobstore, follow the PCF backup process using BBR and copy the external blobstore using your IaaS. Inconsistencies between the blobstore and database may result in apps failing to restart during the restore. You can push these apps again to restart them.
- With PAS configured with an external database and an external blobstore, follow the PCF backup process using BBR, but skip the backup of PAS. Copy the external database and blobstore using your IaaS. Inconsistencies between the blobstore and database may result in apps failing to restart during

the restore. You can push these apps again to restart them.

Best Practices

Frequency of Backups

Pivotal recommends that you take backups in proportion to the rate of change of the data in PCF to minimize the number of changes lost if a restore is required. We suggest starting with backing up every 24 hours. If app developers make frequent changes, you should increase the frequency of backups.

Retention of Backup Artifacts

Operators should retain backup artifacts based on the timeframe they need to be able to restore to. For example, if backups are taken every 24 hours and PCF must be able to be restored to three days prior, three sets of backup artifacts should be retained.

Artifacts should be stored in two data centers other than the PCF data center. When deciding the restore timeframe, you should take other factors such as compliance and auditability into account.

Security

Pivotal strongly recommends that you encrypt artifacts and stored them securely.

Disaster Recovery by Recreating the Deployment

An alternative strategy for recovering PCF after a disaster is to have automation in place so that all the data can be recreated. This requires that every modification to PCF settings and state be automated, typically through use of a pipeline.

Recovery steps include creating a new PCF, recreating orgs, spaces, users, services, service bindings and other state, and re-pushing apps.

For more information about this approach, see the following Cloud Foundry Summit presentation: Multi-DC Cloud Foundry: What, Why and How?.

Disaster Recovery for Different Topologies

Active-Active

To prevent app downtime, some Pivotal customers run active-active, where they run two or more identical PCF deployments in different data centers. If one PCF deployment becomes unavailable, traffic is seamlessly routed to the other deployment. To achieve identical deployments, all operations to PCF are automated so they can be applied to both PCF deployments in parallel.

Because all operations have been automated, the automation approach to disaster recovery is a viable option for active-active. Disaster recovery requires recreating PCF, then running all the automation to recreate state.

This option requires discipline to automate all changes to PCF. Some of the operations that need to be automated are the following:

- App push, restage, scale
- Org, space, and user create, read, update, and delete (CRUD)
- Service instance CRUD
- Service bindings CRUD
- Routes CRUD
- Security groups CRUD
- Quota CRUD

Human-initiated changes always make their way into the system. These changes can include quotas being raised, new settings being enabled, and incident responses. For this reason, Pivotal recommends taking backups even when using an automated disaster recovery strategy.

Using BBR Backup and Restore versus Recreating a Failed PCF Deployment in Active-Active

Disaster Recovery		
	Restore the PCF Data	**Recreate the PCF Data**
Preconditions	IaaS prepared for PCF install	IaaS prepared for PCF install
Steps	1. Recreate PCF 2. Restore 3. Apply changes to make restored PCF match the other active PCF	1. Recreate PCF 2. Trigger automation to recreate orgs, spaces, etc. 3. Notify app developers to repush apps, recreate service instances and bindings
RTO (Recovery Time Objective)		
Platform	Time to recreate PCF	Time to recreate PCF
Apps	Time to restore	Time until orgs/spaces/etc have been recreated + apps have been repushed
RPO (Recovery Point Objective)		
Platform	Time of the last backup	Current time
Apps	Time of the last backup	Current time

Active-Passive

Instead of having a true active-active deployment across all layers, some Pivotal customers prefer to install a PCF or PAS deployment on a backup site. The backup site resides on-premises, in a co-location facility, or the public cloud. The backup site includes an operational deployment, with only the most critical apps ready to accept traffic should a failure occur in the primary data center. Disaster recovery in this scenario involves the following:

1. Switching traffic to the passive PCF, making it active.
2. Recovering the formerly-active PCF. Operators can choose to do this through automation, if that option is available, or by using BBR and the restore process.

The RTO and RPO for recreating the active PCF are the same as outlined in the table above.

Reducing RTO

Both the restore and recreate data disaster recovery options require standing up a new PCF, which can take hours. If you require shorter RTO, several options involving a pre-created standby hardware and PCF are available:

Active-cold	Public cloud environment ready for PCF installation, no PCF installed. This saves both IaaS costs and PCF instance costs. For on-prem installations, this requires hardware on standby, ready to install on, which may not be a realistic option.
Active-warm	PCF installed on standby hardware and kept up to date, VMs scaled down to zero (spin them up each time there is a platform update), no apps installed, no orgs or spaces defined.
Active-inflate platform	Bare minimum PCF install, either with no applications, or a small number of each app in a stopped state. On recovery, push a small number of apps or start current apps, while simultaneously triggering automation to scale the platform to the primary node size, or a smaller version if large percentages of loss are acceptable. This mode allows you to start sending some traffic immediately, while not paying for a full non-primary platform. This method requires data seeded, but it is usually acceptable to complete data sync while platform is scaling up.
Active-inflate apps	Non-primary deployment scaled to the primary node size, or smaller version if large percentages of loss are acceptable, with a small number of Diego cells (VMs). On failover, scale Diego cells up to primary node counts. This mode allows you to start sending most traffic immediately, while not paying for all the AIs of a fully fledged node. This method requires data to be there very quickly after failure. It does not require real-time sync, but near-real time.

There is a tradeoff between cost and RTO: the less the replacement PCF needs to be deployed and scaled, the faster the restore.

Automating Backups

BBR generates the backup artifacts required for PCF, but does not handle scheduling, artifact management, or encryption. The BBR team has created a starter Concourse pipeline to automate backups with BBR.

Also, Stark & Wayne's Shield can be used as a front end management tool using the BBR plugin.

Validating Backups

To ensure that backup artifacts are valid, the BBR tool creates checksums of the generated backup artifacts, and ensures that the checksums match the artifacts on the jumpbox.

However, the only way to be sure that the backup artifact can be used to successfully recreate PCF is to test it in the restore process. This is a cumbersome, dangerous process so should be done with care. For instructions, see Step 11: (Optional) Validate Your Backup of the *Backing Up Pivotal Cloud Foundry with BBR*.

Backing Up Pivotal Cloud Foundry with BBR

Page last updated:

This topic describes the procedure for backing up your critical backend Pivotal Cloud Foundry (PCF) components with BOSH Backup and Restore (BBR), a command-line tool for backing up and restoring BOSH deployments. To restore your backup, see the Restoring Pivotal Cloud Foundry from Backup with BBR topic.

To view the BBR release notes, see the BOSH Backup and Restore Release Notes.

During the backup, BBR stops the Cloud Controller API and the Cloud Controller workers to create a consistent backup. Only the API functionality, like pushing applications or using the Cloud Foundry Command Line Interface (cf CLI) are affected. The deployed applications do not experience downtime.

Note: You can only use BBR to back up PCF v1.11 and later. To back up earlier versions of PCF, perform the manual procedures.

Warnings:
- **Backup artifacts can contain secrets.** Secure backup artifacts by using encryption or other means.
- **The restore is a destructive operation.** BBR is designed to restore PCF after a disaster. If it fails, the environment may be left in an unusable state and require reprovisioning. The two restore scenarios currently documented are Restoring Pivotal Cloud Foundry from Backup with BBR and Restoring a PAS Backup In-Place.
- **The Cloud Controller API will stop sending and receiving calls** for the duration of PCF restoration.
- **BBR does not back up any service data.**
- **Using an external blobstore or an external MySQL database results in an unusable restore** even if the restore appears to run successfully.
- **For PCF v2.0.0, BBR only supports backup and restore of environments with zero or one CredHub instances.**

Recommendations

Pivotal recommends the following:

- Follow the full procedure documented in this topic when creating a backup. This ensures that you always have a consistent backup of Ops Manager and PAS to restore from.
- Back up frequently, especially before making any changes to your PCF deployment, such as the configuration of any tiles in Ops Manager.

Compatibility of Restore

When using a backup to restore, you must ensure that the restore environment is compatible. For more information, see Compatibility of Restore.

Supported Components

BBR is a binary that can back up and restore BOSH deployments and BOSH Directors. BBR requires that the backup targets supply scripts that implement the backup and restore functions.

BBR backs up the following PCF components:

- **PAS**: PAS must be configured with an internal MySQL database and a WebDAV/NFS blobstore to be backed up and restored with BBR. BBR does not support PAS with an external blobstore or an external MySQL database.

  Warning: The following guidelines apply when you are using BBR to back up PCF and have configured an external blobstore and/or external database in PAS:

  - If you configured an internal database and an external blobstore in PAS, then follow the PCF backup process, and copy the external blobstore via the IaaS. There may be inconsistencies between the blobstore and database, which may result in some apps not coming up during a restore. Those apps can be repushed.
  - If you configured an external database and an external blobstore in PAS, then follow the PCF backup process, but skip the backup of PAS. Copy the external database and blobstore via the IaaS. There may be inconsistencies between the blobstore and database, which may result

in some apps not coming up during a restore. Those apps can be repushed.

- **BOSH Director**: The BOSH Director must have an internal Postgres database to be backed up and restored with BBR. As part of backing up the BOSH Director, BBR backs up the BOSH UAA database and the CredHub database.

Backing Up Services

Warning: BBR does **not** currently back up any service data.

Keep in mind the following when backing up services:

- You can back up and restore brokered services with the procedures documented in this topic and in the Restoring Pivotal Cloud Foundry from Backup with BBR topic. BBR backs up and restores the VMs and the service instances, but not the service data.
- You can redeploy on-demand service instances manually during restore, but the data in the instance is not backed up.
- BBR does not back up managed services or their data.

Workflow

Operators download the BBR binary and transfer it to a jumpbox. Then they run BBR from the jumpbox, specifying the name of the BOSH deployment to back up.

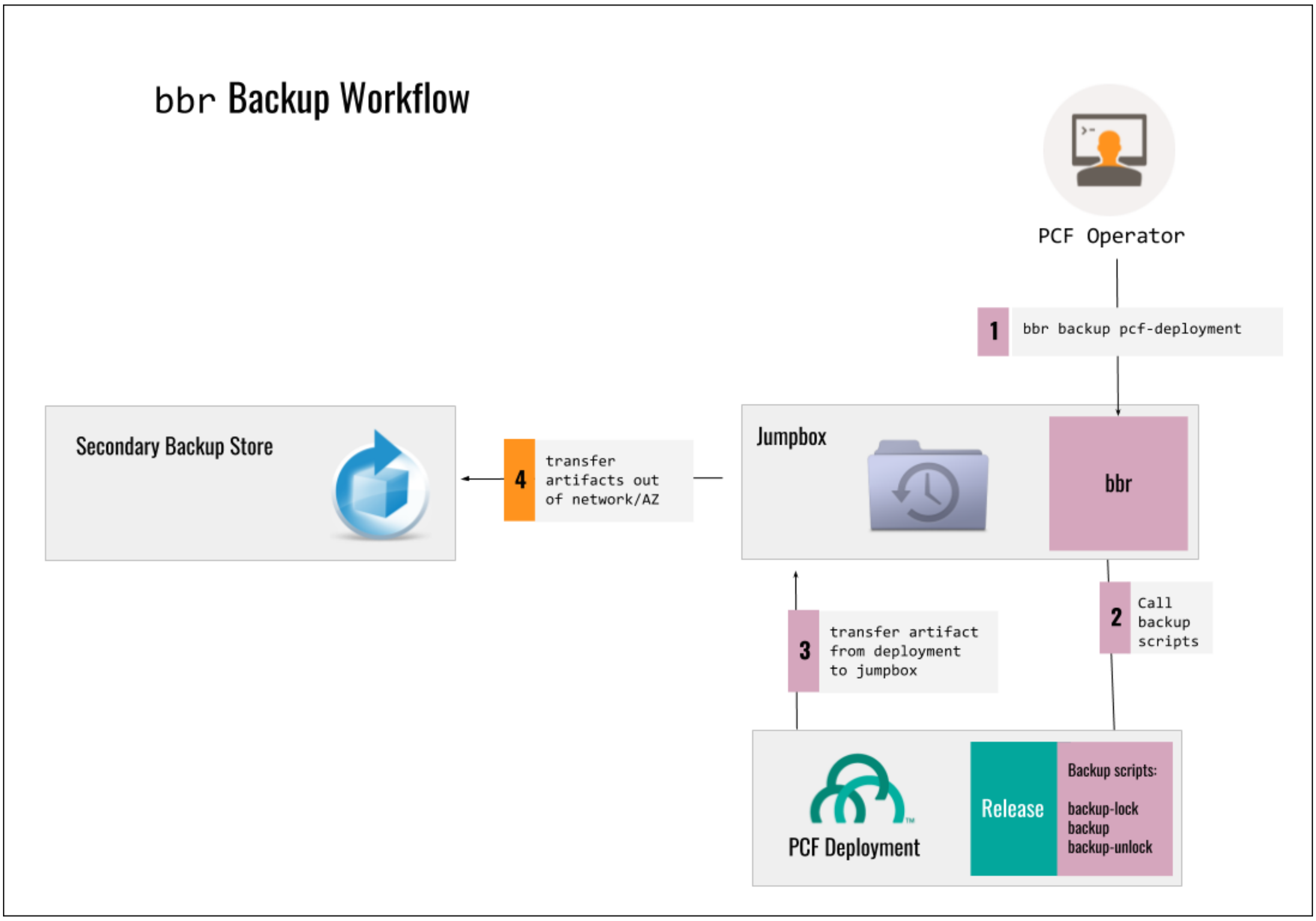
BBR examines the jobs in the BOSH deployment, and triggers the scripts in the following stages:

1. **Pre-backup lock**: The pre-backup lock scripts locks the job so backups are consistent across the cluster.
2. **Backup**: The backup script backs up the release.
3. **Post-backup unlock**: The post-backup unlock script unlocks the job after the backup is complete.

Scripts in the same stage are all triggered together. For instance, BBR triggers all pre-backup lock scripts before any backup scripts. Scripts within a stage may be triggered in any order.

The backup artifacts are drained to the jumpbox, where the operator can transfer them to storage and use them to restore PCF.

The following diagram shows a sample backup flow.



Before using BBR, follow the instructions in the Setting up your system for BBR section.

Prepare to Create Your Backup

Step 1: Set Up Your Jumpbox

Prepare your jumpbox for BBR by following the steps in the Setting Up Your Jumpbox for BBR topic.

Step 2: Record the Cloud Controller Database Encryption Credentials

Perform the following steps to retrieve the Cloud Controller Database encryption credentials from the PAS tile:

1. Navigate to Ops Manager in a browser and log in to the Ops Manager Installation Dashboard.
2. Select PAS **Credentials** and locate the Cloud Controller section.
3. Record the Cloud Controller **DB Encryption Credentials**. You must provide these credentials if you contact Pivotal Support for help restoring your installation.

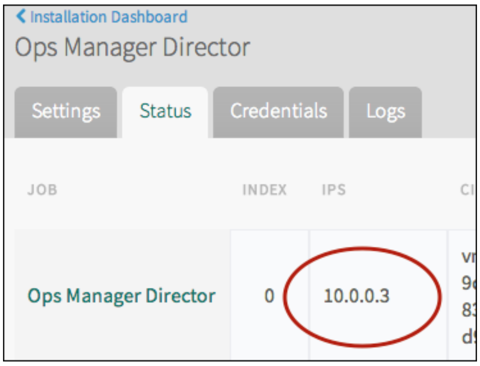
Cloud Controller	VM Credentials	Link to Credential
	Staging Upload Credentials	Link to Credential
	Bulk Api Credentials	Link to Credential
	Db Encryption Credentials	Link to Credential
	Encrypt Key	Link to Credential

Step 3: Retrieve BOSH Director Address and Credentials

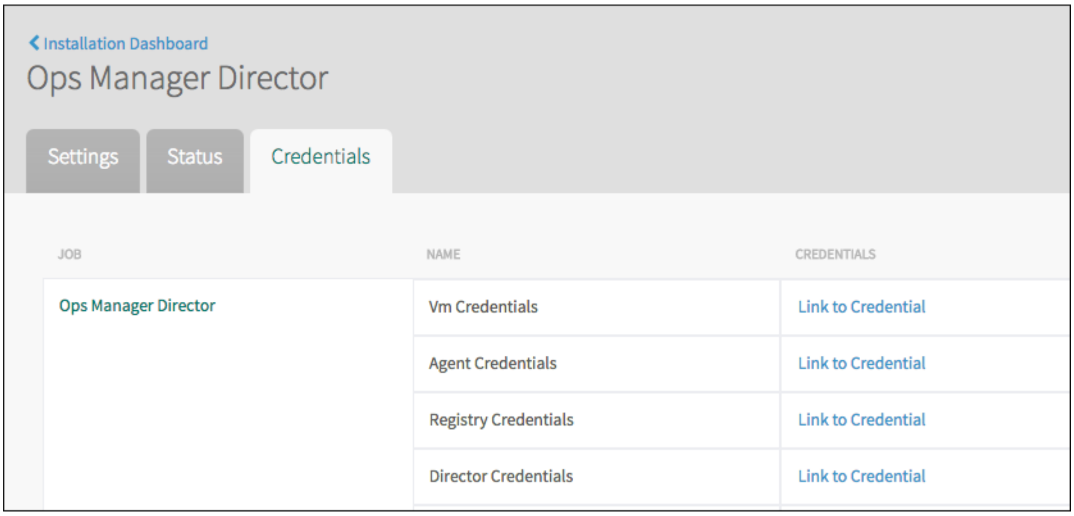
Perform the following steps to retrieve the IP address of your BOSH Director and the credentials for logging in from the Ops Manager Director tile:

1. Install the BOSH v2 CLI on a machine outside of your PCF deployment. You can use the jumpbox for this task.

2. From the Installation Dashboard in Ops Manager, select **Ops Manager Director > Status** and record the IP address listed for the Director. You access the BOSH Director using this IP address.



3. Click **Credentials** and record the Director credentials.



4. From the command line, log into the BOSH Director using the IP address and credentials that you recorded:

```
$ bosh -e DIRECTOR_IP \
--ca-cert PATH-TO-BOSH-SERVER-CERT log-in
Email (): director
Password (): *******************
Successfully authenticated with UAA
Succeeded
```

Step 4: Check your BOSH Director

Perform the following steps to back up your BOSH Director:

1. Navigate to the Ops Manager Installation Dashboard.

2. Click the Ops Manager tile.

3. Click the **Credentials** tab.

4. Locate **Bbr Ssh Credentials** and click **Link to Credential** next to it.

   You can also retrieve the credentials using the Ops Manager API with a GET request to the following endpoint: `/api/v0/deployed/director/credentials/bbr_ssh_credentials` . For more information, see the Using the Ops Manager API topic.

5. Copy the value for `private_key_pem`.

6. SSH into your jumpbox:

```
$ ssh JUMPBOX-USER/JUMPBOX-ADDRESS -i YOUR-CERTIFICATE.pem
```

7. Run the following command to reformat the key and save it to a file named `PRIVATE-KEY` in the current directory, pasting in the contents of your private key for `YOUR-PRIVATE-KEY`:

```
$ printf -- "YOUR-PRIVATE-KEY" > PRIVATE-KEY
```

8. Run the BBR `pre-backup-check` command from your jumpbox:

```
$ bbr director \
 --private-key-path PRIVATE-KEY \
 --username bbr \
 --host HOST \
 pre-backup-check
```

   Use the optional `--debug` flag to enable debug logs. See the Logging section for more information.

   Replace the placeholder values as follows:
   - `PRIVATE-KEY`: This is the path to the private key file you created above.
   - `HOST`: This is the address of the BOSH Director. If the BOSH Director is public, this is a URL, such as `https://my-bosh.xxx.cf-app.com`. Otherwise, this is the `BOSH-DIRECTOR-IP`, which you retrieved in the Step 3: Retrieve BOSH Director Address and Credentials section.

9. If the pre-backup check succeeds, continue to the next section. If it fails, the Director may not have the correct backup scripts, or the connection to the BOSH Director may have failed.

Step 5: Identify Your Deployment

After logging in to your BOSH Director, run the following command to identify the name of the BOSH deployment that contains PCF:

```
$ bosh -e DIRECTOR-IP --ca-cert PATH-TO-BOSH-SERVER-CERTIFICATE deployments

Name          Release(s)
cf-example      push-apps-manager-release/661.1.24
           cf-backup-and-restore/0.0.1
           binary-buildpack/1.0.11
           capi/1.28.0
           cf-autoscaling/91
           cf-mysql/35
           ...
```

In the above example, the name of the BOSH deployment that contains PCF is `cf-example`.

Step 6: Check Your Deployment

Perform the following steps to check that your BOSH Director is reachable and has a deployment that can be backed up:

1. From your jumpbox, run the BBR pre-backup check:

```
$ BOSH-CLIENT-SECRET=BOSH-PASSWORD \
 bbr deployment \
 --target BOSH-DIRECTOR-IP \
 --username BOSH-CLIENT \
 --deployment DEPLOYMENT-NAME \
 --ca-cert PATH-TO-BOSH-SERVER-CERTIFICATE \
 pre-backup-check
```

   Replace the placeholder values as follows:

   - `BOSH-CLIENT`, `BOSH-PASSWORD`: From the Ops Manager Installation Dashboard, click **Ops Manager Director**, navigate to the **Credentials** tab, and click **Uaa Bbr Client Credentials** to retrieve the BOSH UAA credentials.

     You can also retrieve the credentials using the Ops Manager API with a GET request to the following endpoint:

`/api/v0/deployed/director/credentials/uaa_bbr_client_credentials` . For more information, see the Using the Ops Manager API topic.

- `BOSH-DIRECTOR-IP` : You retrieved this value in the Step 3: Retrieve BOSH Director Address and Credentials section.
- `DEPLOYMENT-NAME` : You retrieved this value in the Step 6: Identify Your Deployment section.
- `PATH-TO-BOSH-SERVER-CERTIFICATE` : This is the path to the BOSH Director's Certificate Authority (CA) certificate, if the certificate is not verifiable by the local machine's certificate chain. If you are using the Ops Manager VM as your jumpbox, locate the certificate at `/var/tempest/workspaces/default/root_ca_certificate` .

2. If the pre-backup check succeeds, continue to the next section. If it fails, the deployment you selected may not have the correct backup scripts, or the connection to the BOSH Director may have failed.

Create Your Backup

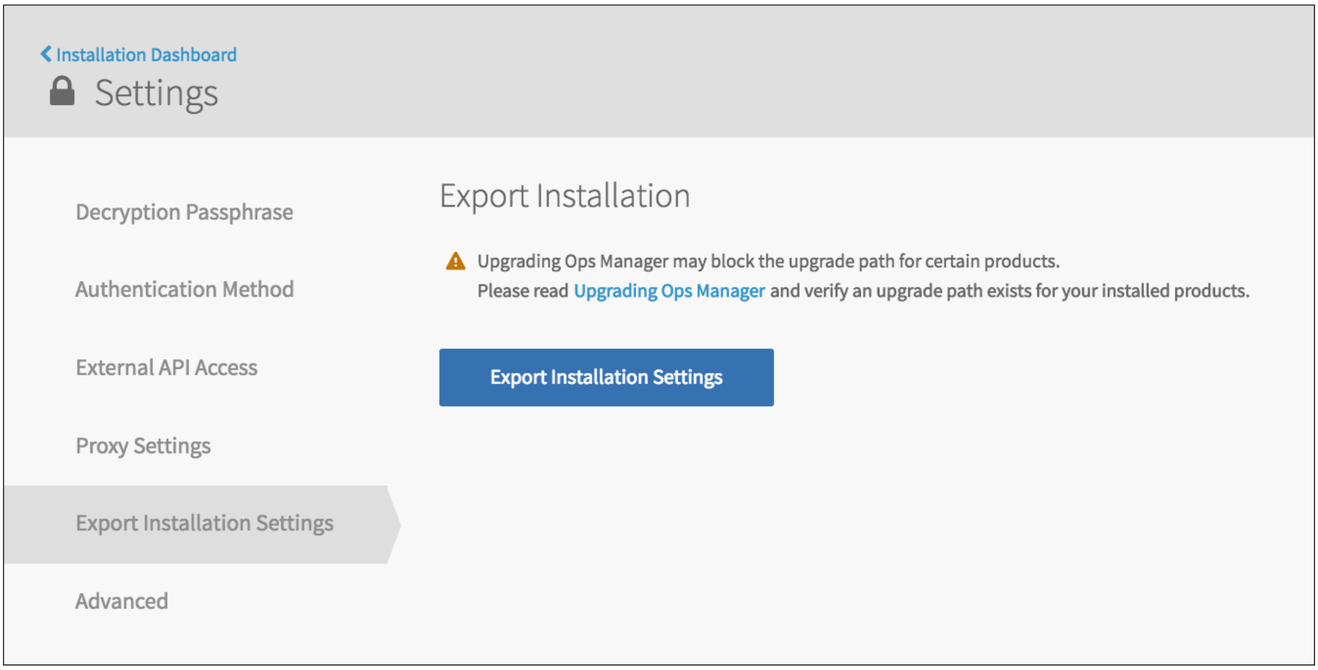
Step 7: Export Installation Settings

Pivotal recommends that you back up your Ops Manager installation settings by exporting frequently. This option is only available after you have deployed at least one time. Always export your installation settings before following the steps in the Import Installation Settings section of the Restoring Pivotal Cloud Foundry from Backup with BBR topic.

> **Note**: Exporting your installation settings only backs up the settings you configured in Ops Manager. It does not back up your virtual machines (VMs) or any external MySQL databases.

From the **Installation Dashboard** in the Ops Manager interface, click your user name at the top right navigation. Select **Settings**.

> **Note**: Ops Manager 1.12 exports installation settings only, so the output is much smaller than in previous Ops Manager versions.



Step 8: Back Up Your BOSH Director

Run the BBR backup command from your jumpbox to back up your BOSH Director:

```
$ bbr director \
  --private-key-path PRIVATE-KEY \
  --username bbr \
  --host HOST \
  backup
```

Use the optional `--debug` flag to enable debug logs. See the Logging section for more information.

Replace the placeholder values as follows:

- `PRIVATE-KEY` : This is the path to the private key file you created in Step 4: Check your BOSH Director.
- `HOST` : This is the address of the BOSH Director. If the BOSH Director is public, this is a URL, such as `https://my-bosh.xxx.cf-app.com` . Otherwise, this is the `BOSH-DIRECTOR-IP` , which you retrieved in the Step 3: Retrieve BOSH Director Address and Credentials section.

Notes:
- The BOSH Director backup takes at least 20 minutes.
- Because the BBR backup command can take a long time to complete, Pivotal recommends you run it independently of the SSH session so that the process can continue running even if your connection to the jumpbox fails.
  Use `nohup` , a `screen` , or a `tmux` session.

If your backup terminates early or fails, you can clean up by running backup-cleanup:

```
$ bbr director \
 --private-key-path PRIVATE-KEY \
 --username bbr \
 --host HOST \
 backup-cleanup
```

Step 9: Back Up Your PAS Deployment

1. If you are using an external blobstore, create a copy of the blobstore with your IaaS specific tool. Your blobstore backup may be slightly inconsistent with your PAS backup depending on the duration of time between performing the backups.

2. If you are using an external blobstore, create a copy of the blobstore with your IaaS specific tool. Your blobstore backup may be slightly inconsistent with your PAS backup depending on the duration of time between performing the backups.

3. Run the BBR backup command from your jumpbox to back up your PAS deployment:

   ```
   $ BOSH-CLIENT-SECRET=BOSH-PASSWORD \
   nohup bbr deployment \
   --target BOSH-DIRECTOR-IP \
   --username BOSH-CLIENT \
   --deployment DEPLOYMENT-NAME \
   --ca-cert PATH-TO-BOSH-SERVER-CERTIFICATE \
   backup
   ```

   - Use the optional `--debug` flag to enable debug logs. See the Logging section for more information.
   - Use the optional `--with-manifest` flag to ensure that BBR downloads your current deployment manifest when backing up. These manifests are included in the Ops Manager export, but are useful for reference. For example:

     ```
     $ BOSH-CLIENT-SECRET=BOSH-PASSWORD \
     nohup bbr deployment \
     --target BOSH-DIRECTOR-IP \
     --username BOSH-CLIENT \
     --deployment DEPLOYMENT-NAME \
     --ca-cert PATH-TO-BOSH-SERVER-CERTIFICATE \
     backup --with-manifest
     ```

Notes:
- Backing up PAS takes at least 10 minutes and can take considerably longer with larger blobstores or slow network connections. The backup also incurs around 10 minutes of Cloud Controller downtime during which users are unable to push, scale, or delete apps. Your apps will not be affected.
- Because the BBR backup command can take a long time to complete, Pivotal recommends you run it independently of the SSH session, so that the process can continue running even if your connection to the jumpbox fails. The command above uses `nohup` but you could also run the command in a `screen` or `tmux` session.
- The backup artifact will contain one or more empty UAA tar files. This is an extraneous artifact. The UAA backup is included in the MySQL backup.

If the command fails, do the following:

1. Run backup-cleanup:

```
$ BOSH-CLIENT-SECRET=BOSH-PASSWORD \
 bbr deployment \
--target BOSH-DIRECTOR-IP \
--username BOSH-CLIENT \
--deployment DEPLOYMENT-NAME \
--ca-cert PATH-TO-BOSH-SERVER-CERTIFICATE \
backup-cleanup
```

2. Ensure all the parameters in the command are set.

3. Ensure the BOSH Director credentials are valid.

4. Ensure the specified deployment exists.

5. Consult the Exit Codes section below.

After Taking the Backups

If the commands complete successfully, perform the following steps:

1. Move the backup artifacts off the jumpbox to your preferred storage space. The backup created by BBR consists of a folder with the backup artifacts and metadata files. Pivotal recommends compressing and encrypting the files.

2. Make redundant copies of your backup and store them in multiple locations to minimize the risk of losing backups in the event of a disaster.

3. Attempt to test restore on every backup to validate it. Perform the procedures in the next step. See Step 11: Validate Your Backup.

Step 10: (Optional) Validate Your Backup

If you want to validate your backup, follow the instructions that correspond to your use case:

- **Production environments**: Validate Your Entire Backup
- **Sandbox and other non-production environments**: Validate Only Your PAS Backup

Validate Your Entire Backup

Warnings:
- **The VMs and disks from the backed-up BOSH Director should not be visible to the new BOSH Director** when validating your backup. As a result, Pivotal recommends that you deploy the new BOSH Director to a different IaaS network and account than the VMs and disks of the backed-up BOSH Director.
- **Data services outside of PCF may produce unexpected side effects during restoration.** The services will try to connect to those data services when you restore to a new environment. For example, consider an app that processes mail queues and connects to an external database. When you validate your backup in a test environment, the app may start processing the queue, and this work may be lost.

After backing up PCF, you may want to validate your backup by restoring it to a similar environment and checking the applications. Because BBR is designed for disaster recovery, its backups are intended to be restored to an environment deployed with the same configuration.

Perform the following steps to spin up a second environment that matches the original in order to test a restore:

1. Export your Ops Manager installation by performing the steps in the Step 8: Export Installation Settings section.

2. Create a new Ops Manager VM in a different network to the original. Ensure that the Ops Manager VM has enough persistent disk to accommodate the files exported in the previous step. Consult the topic specific to your IaaS:

    - Deploying BOSH and Ops Manager to AWS
    - Manually Configuring AWS for PCF
    - Launching an Ops Manager Director Instance with an ARM Template
    - Launching an Ops Manager Director Instance on GCP
    - Provisioning the OpenStack Infrastructure

- Deploying Operations Manager to vSphere

After you deploy the second environment, follow the instructions in the Restoring Pivotal Cloud Foundry from Backup with BBR topic.

Validate Your PAS Backup Only

For a sandbox or other non-production environment, you can optionally perform an *in-place* restore of PAS only. In this case, you restore the PAS backup to the same PCF environment that the backup was created from. Follow the procedures in Restoring an PAS Backup In-place.

Exit Codes and Logging

For information about the exit codes returned by BBR and BBR logging, consult the sections below.

Exit Codes

The exit code returned by BBR indicates the status of the backup. The following table matches exit codes to error messages.

Value	Error
0	Success
1	General failure
4	The pre-backup lock failed.
8	The post-backup unlock failed. Your deployment may be in a bad state and requires attention.
16	The cleanup failed. This is a non-fatal error indicating that the utility has been unable to clean up open BOSH SSH connections to the deployment VMs. Manual cleanup may be required to clear any hanging BOSH users and connections.

If multiple failures occur, your exit code reflects a combination of values. Use bitwise AND to determine which failures occurred.

For example, the exit code `5` indicates that the pre-backup lock failed and a general error occurred.

To check that a bit is set, use bitwise AND, as demonstrated by the following example of exit code `20`:

```
20 | 1  == 1   # false
20 | 4  == 4   # true; lock failed
20 | 8  == 8   # false
20 | 16 == 16  # true; cleanup failed
```

Exit code `20` indicates that the pre-backup lock failed and cleanup failed.

Logging

BBR outputs logs to stdout. By default, BBR logs:

- The backup and restore scripts that it finds
- When it starts or finishes a stage, such as pre-backup scripts or backup scripts
- When the process is complete
- When any error occurs

If more logging is needed, use the optional `--debug` flag to print the following information:

- Logs about the API requests made to the BOSH server
- All commands executed on remote instances
- All commands executed on local environment
- Standard in and standard out streams for the backup and restore scripts when they are executed

Cancelling a Backup

If you need to cancel a backup, perform the following steps:

1. Terminate the BBR process by pressing Ctrl-C and then typing `yes` to confirm.
2. Log in to your BOSH Director with the BOSH CLI by performing the procedures in the Step 3: Retrieve BOSH Director Address and Credentials section above.
3. Perform the following steps for each `cloud_controller` VM in your deployment:
   a. List the VMs in your deployment:
   ```
   $ bosh -e DIRECTOR-IP --ca-cert /var/tempest/workspaces/default/root_ca_certificate \
   -d DEPLOYMENT-NAME \
   ssh
   ```
   b. Select the VM you want to SSH into.
   c. Run the following command from the VM:
   ```
   $ sudo /var/vcap/jobs/cloud-controller-backup/bin/bbr/post-backup-unlock
   ```
4. Run the BBR pre-backup check from your jumpbox by following the steps in the Step 7: Check Your Deployment section above. If the command reports that it cannot back up the deployment, SSH onto each VM mentioned in the error using the BOSH CLI and remove the `/var/vcap/store/bbr-backup` directory if present.

PAS Component Availability During Backup

This topic describes the operational impact of backing up Pivotal Application Service (PAS) components with BBR. To ensure correctness of backups, each component that requires backup has its own set of scripts. The sections in this topic describe the availability of each component during backup of its database.

Cloud Controller

The Cloud Controller is unavailable during backup. Apps and Services continue to run as normal, but you cannot perform operations that require the Cloud Controller API. This includes the following:

- Pushing new apps or creating new services
- Modifying existing apps or services
- Using the clients of the Cloud Controller API, such as the following:
  - The Cloud Foundry Command Line Interface (cf CLI)
  - Apps Manager and its integrations
  - The Java client used by Spring apps

The backup process for the Cloud Controller is as follows:

Stage	Description
1: Pre-backup lock	The processes running on the **Cloud Controller Worker**, **Cloud Controller**, and **Clock Global** VMs are stopped.
2: Backup	The BBR SDK backup script runs to backup the Cloud Controller database (CCDB), which contains state information for apps on your deployment.
3: Post-backup unlock	The processes start again on the **Cloud Controller Worker**, **Cloud Controller**, and **Clock Global** VMs.

UAA

UAA remains available in read-only mode during backup. This means that you cannot perform write operations for clients, users, groups, identity providers, or zone configuration. However, you can continue performing read operations, such as generating, validating, and revoking tokens. Additionally, UAA continues to authenticate users and authorize requests for users and clients.

The read-only behavior during backup applies to all of the following ways of accessing UAA: the UAA API, the UAA CLI, cf CLI, login screens, and services such as the Single Sign-On Service tile.

The backup process for UAA is as follows:

Stage	Description
1: Pre-backup lock	UAA enters read-only mode.
2: Backup	The BBR SDK backup script runs to backup the UAA database, which contains Cloud Foundry user credentials.
3: Post-backup unlock	UAA exits read-only mode.

Routing API

The Routing API remains available during backup. However, you cannot perform write operations using the Routing API because the routing database is locked. All read operations offered by the Routing API remain available.

The BBR SDK backup script for the Routing API backs up its database, which contains router groups, routes, and internal implementation information.

Usage Service

The Usage Service is unavailable during backup. You cannot access the API as described in Monitoring App, Task, and Service Instance Usage. Additionally, you cannot view usage and accounting reports as described in Monitoring Instance Usage with Apps Manager.

The backup process for Usage Service is as follows:

Note: The Usage Service runs as a set of Cloud Foundry apps in the `system` org.

Stage	Description
1: Pre-backup lock	The Usage Service apps in the `system` org stop. This lock occurs before the Cloud Controller and UAA components lock.
2: Backup	The BBR SDK backup script runs to backup the Usage Service database.
3: Post-backup unlock	The Usage Service apps in the `system` org start again. This unlock occurs after the Cloud Controller and UAA components unlock.

App Autoscaler

The App Autoscaler service is unavailable during backup. You cannot access the UI or API. For any apps configured to use the App Autoscaler, the service does not scale these apps during backup.

The backup process for App Autoscaler is as follows:

Note: The App Autoscaler service runs as a set of Cloud Foundry apps in the `system` org.

Stage	Description
1: Pre-backup lock	The Autoscaler apps in the `system` org stop. This lock occurs before the Cloud Controller and UAA components lock.
2: Backup	The BBR SDK backup script runs to backup the App Autoscaler database.
3: Post-backup unlock	The Autoscaler apps in the `system` org start again. This unlock occurs after the Cloud Controller and UAA components unlock.

NFS Volume Service

The NFS service broker backup scripts rely on the locking of the Cloud Controller to stop traffic to its service. This is because the Cloud Controller is responsible for invoking the NFS service broker.

When the Cloud Controller locks during backup, you cannot create or delete new instances or bindings of a volume service. However, apps already bound to a volume service continue to operate normally during backups.

The NFS service broker backup script performs a backup of the database used to store service instances and service bindings for the NFS service broker.

Notification Service

The Notification Service is not available during backup with BBR due to its dependency on the Cloud Controller. Notifications cannot be sent while the Cloud Controller is unavailable.

Network Policy Server

The Network Policy Server is unavailable during backup. While existing policies are still enforced, you cannot use the cf CLI to add or remove policies for Container Networking as documented in Administering Container-to-Container Networking.

CredHub

PAS CredHub is unavailable during backup. If the service instance credentials for an app are stored in CredHub, the app cannot fetch those credentials during backup. In some cases, apps may not start if they cannot fetch credentials for a service instance binding.

Restoring PCF from Backup with BBR

Page last updated:

This topic describes the procedure for restoring your critical backend Pivotal Cloud Foundry (PCF) components with BOSH Backup and Restore (BBR), a command-line tool for backing up and restoring BOSH deployments. To perform the procedures in this topic, you must have backed up PCF by following the steps in the Backing Up Pivotal Cloud Foundry with BBR topic.

To view the BBR release notes, see BOSH Backup and Restore Release Notes.

The procedures described in this topic prepare your environment for PCF, deploy Ops Manager, import your installation settings, and use BBR to restore your PCF components.

Warning: Restoring PCF with BBR is a destructive operation. If the restore fails, the new environment may be left in an unusable state and require reprovisioning. Only perform the procedures in this topic for the purpose of disaster recovery, such as recreating PCF after a storage-area network (SAN) corruption.

Warning: When validating your backup, the VMs and disks from the backed-up BOSH Director should not visible to the new BOSH Director. As a result, Pivotal recommends that you deploy the new BOSH Director to a different IaaS network and account than the VMs and disks of the backed up BOSH Director.

Warning: For PCF v2.0.0, BBR only supports backup and restore of environments with zero or one CredHub instances.

Note: BBR is a feature in PCF v1.11. You can only use BBR to back up PCF v1.11 and later. To restore earlier versions of PCF, perform the manual procedure documented for your specific PCF version.

Note: If you are restoring in order to validate a backup, look for notes marked **Validation** throughout the topic.

Compatibility of Restore

This section describes the restrictions for a backup artifact to be restorable to another environment. This section is for guidance only, and Pivotal highly recommends that operators validate their backups by using the backup artifacts in a restore.

Consult the following restrictions for a backup artifact to be restorable:

- **Topology**: BBR requires the BOSH topology of a deployment to be the same in the restore environment as it was in the backup environment.
- **Naming of instance groups and jobs**: For any deployment that implements the backup and restore scripts, the instance groups and jobs must have the same names.
- **Number of instance groups and jobs**: For instance groups and jobs that have backup and restore scripts, there must be the same number of instances.
- **Limited validation**: BBR puts the backed up data into the corresponding instance groups and jobs in the restored environment, but can't validate the restore beyond that. For example, if the MySQL encryption key is different in the restore environment, the BBR restore might succeed although the restored MySQL database is unusable.
- **PCF version**: BBR can restore to the same version of PCF that was backed up. BBR does not support restoring to other major, minor, or patch releases.

Note: A change in VM size or underlying hardware should not affect BBR's ability to restore data, as long as there is adequate storage space to restore the data.

(Optional) Step 1: Prepare Your Environment

In an event of a disaster, you may lose not only your VMs and disks, but your IaaS resources as well, such as networks and load balancers.

If you need to recreate your IaaS resources, prepare your environment for PCF by following the instructions specific to your IaaS in Installing Pivotal Cloud Foundry.

Note: The instructions for installing PCF on Amazon Web Services (AWS) and OpenStack combine the procedures for preparing your environment and deploying Ops Manager into a single topic. The instructions for the other supported IaaSes split these procedures into two separate topics.

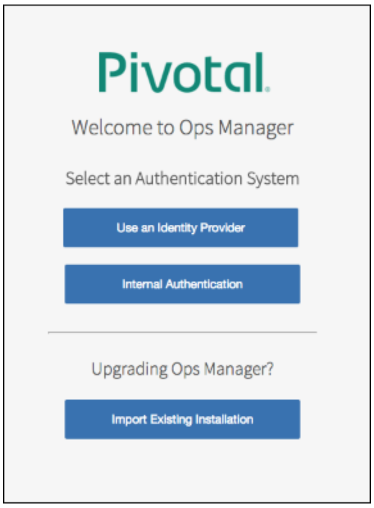
If you recreate your IaaS resources, you must also add those resources to Ops Manager by performing the procedures in the (Optional) Step 3: Configure Ops Manager for New Resources section.

Step 2: Deploy Ops Manager and Import Installation Settings

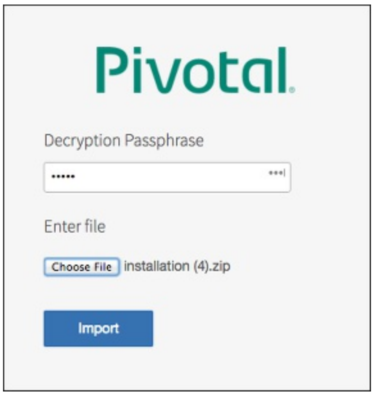
1. Perform the procedures for your IaaS to deploy Ops Manager:
    - If you configured AWS manually: Step 12 through Step 19 of the *Manually Configuring AWS for PCF* topic.
    - If you used the CloudFormation template install PCF on AWS: Deploying BOSH and Ops Manager to AWS
    - Launching an Ops Manager Director Instance on Azure
    - Launching an Ops Manager Director Instance on GCP
    - Step 3 through Step 7 of the *Provisioning the OpenStack Infrastructure* topic.
    - Deploying Operations Manager to vSphere

2. Access your new Ops Manager by navigating to `YOUR-OPS-MAN-FQDN` in a browser.

3. On the **Welcome to Ops Manager** page, click **Import Existing Installation**.



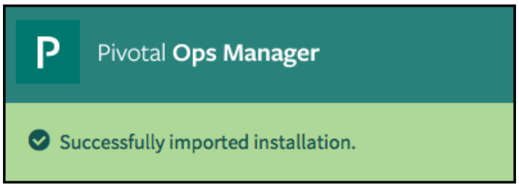
4. In the import panel, perform the following tasks:
    - Enter your **Decryption Passphrase**.
    - Click **Choose File** and browse to the installation zip file that you exported in the Step 7: Export Installation Settings section of the *Backing Up Pivotal Cloud Foundry with BBR* topic.



5. Click **Import**.

> **Note**: Some browsers do not provide feedback on the status of the import process, and may appear to hang. The import process takes at least 10 minutes, and takes longer the more tiles that were present on the backed-up Ops Manager.

6. A **Successfully imported installation** message appears upon completion.



(Optional) Step 3: Configure Ops Manager for New Resources

If you recreated IaaS resources such as networks and load balancers by following the steps in the (Optional) Step 1: Prepare Your Environment section above, perform the following steps to update Ops Manager with your new resources:

1. Enable Ops Manager advanced mode by following the instructions in this Knowledge Base article.
2. Navigate to the Ops Manager Installation Dashboard and click the Ops Manager Director tile.
3. Click **Create Networks** and update the network names to reflect the network names for the new environment.
4. If running on GCP, click **Google Config** and update the **Project ID** to reflect the new GCP project ID.
5. Return to the Ops Manager Installation Dashboard and click the Pivotal Application Service (PAS) tile.
6. Click **Resource Config**. If necessary for your IaaS, enter the name of your new load balancers in the **Load Balancers** column.
7. If necessary, click **Networking** and update the load balancer SSL certificate and private key under **Router SSL Termination Certificate and Private Key**.
8. If your environment has a new DNS address, update the old environment DNS entries to point to the new load balancer addresses. For more information, see the Step 4: Configure Networking section of the *Using Your Own Load Balancer* topic and follow the link to the instructions for your IaaS.
9. If you are using Google Cloud Platform (GCP), navigate to the **Google Config** section of the Ops Manager Director tile and update the **Default Deployment Tag** to reflect the new environment.
10. Disable Ops Manager advanced mode as recommended in the Knowledge Base article.

Step 4: Remove BOSH State File

1. SSH into your Ops Manager VM. For more information, see the SSH into Ops Manager section of the *Advanced Troubleshooting with the BOSH CLI* topic.
2. On the Ops Manager VM, delete the `/var/tempest/workspaces/default/deployments/bosh-state.json` file:

```
$ sudo rm /var/tempest/workspaces/default/deployments/bosh-state.json
```

3. Navigate to `YOUR-OPS-MAN-FQDN` in a browser and log into Ops Manager.
4. Upload the required stemcell for each tile that requires one. Each tile that requires an upload displays with an orange indicator at the bottom. When clicked, the Stemcell panel shows an orange circle beside it.

   Warning: Do not click **Apply Changes** at this point.

Step 5: Deploy Ops Manager Director

Perform the steps in the Applying Changes to Ops Manager Director topic to use the Ops Manager API to only deploy the Ops Manager Director.

Validation: If your BOSH Director has an external hostname, you should change it in **Ops Manager Director > Director Config > Directory Hostname** to ensure it does not conflict with the hostname of the backed up Director.

Step 6: Transfer Artifacts to Jumpbox

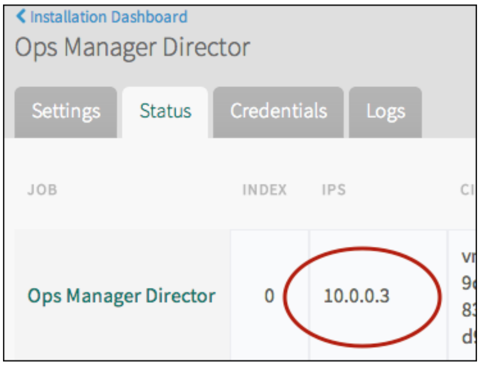
In the Step 9: Back Up Your PAS Deployment section of the *Backing Up Pivotal Cloud Foundry with BBR* topic, in the *After Taking the Backups* section you moved the TAR and metadata files of the backup artifacts off your jumpbox to your preferred storage space. Now you must transfer those files back to your jumpbox.

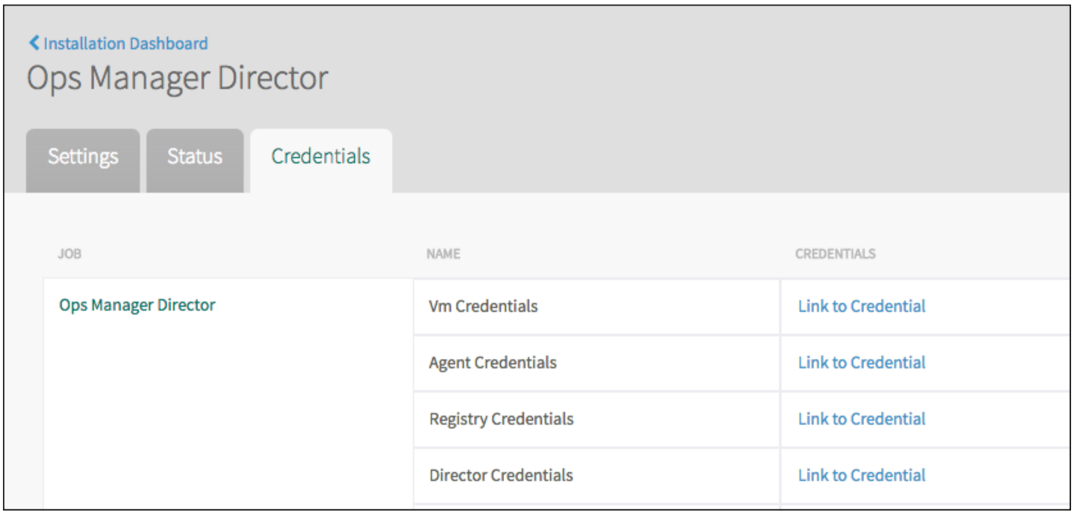
For instance, you could SCP the backup artifact to your jumpbox:

```
$ scp LOCAL-PATH-TO-BACKUP-ARTIFACT JUMPBOX-USER/JUMPBOX-ADDRESS
```

Step 7: Retrieve BOSH Director Address and Credentials

Perform the following steps to retrieve the IP address of your BOSH Director and the credentials for logging in from the Ops Manager Director tile:

1. Install the BOSH v2 CLI on a machine outside of your PCF deployment. You can use the jumpbox for this task.
2. From the Installation Dashboard in Ops Manager, select **Ops Manager Director > Status** and record the IP address listed for the Director. You access 

the BOSH Director using this IP address.
3. Click **Credentials** and record the Director credentials.


4. From the command line, log into the BOSH Director using the IP address and credentials that you recorded:

```
$ bosh -e DIRECTOR_IP \
--ca-cert PATH-TO-BOSH-SERVER-CERT log-in
Email (): director
Password (): ******************
Successfully authenticated with UAA
Succeeded
```

Step 8: Restore the BOSH Director

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the Ops Manager tile.
3. Click the **Credentials** tab.
4. Locate **Bbr Ssh Credentials** and click **Link to Credential** next to it.

   You can also retrieve the credentials using the Ops Manager API with a GET request to the following endpoint: `/api/v0/deployed/director/credentials/bbr_ssh_credentials`. For more information, see the Using the Ops Manager API topic.
5. Copy the value for `private_key_pem`.
6. SSH into your jumpbox.
7. Run the following command to reformat the key and save it to a file named `PRIVATE-KEY` in the current directory, copying in the contents of your private key for `YOUR-PRIVATE-KEY`:

   ```
   $ printf -- "YOUR-PRIVATE-KEY" > PRIVATE-KEY
   ```
8. Ensure the BOSH Director backup artifact is in the folder you from which you will run BBR.
9. Run the BBR restore command from your jumpbox to restore the BOSH Director:

   ```
   $ nohup bbr director \
     --private-key-path PRIVATE-KEY \
     --username bbr \
     --host HOST \
     restore \
       --artifact-path PATH-TO-DIRECTOR-BACKUP
   ```

   Use the optional `--debug` flag to enable debug logs. See the Logging section of the *Backing Up Pivotal Cloud Foundry with BBR* topic for more information.

   Replace the placeholder values as follows:
   - `PATH-TO-DIRECTOR-BACKUP`: This is the path to the Director backup you want to restore.
   - `PRIVATE-KEY`: This is the path to the private key file you created above.
   - `HOST`: This is the address of the BOSH Director. If the BOSH Director is public, this is a URL, such as `https://my-bosh.xxx.cf-app.com`. Otherwise, it is the `BOSH-DIRECTOR-IP`, which you retrieved in the Step 6: Retrieve BOSH Director Address and Credentials section.

Note: The BBR BOSH Director restore command can take at least 15 minutes to complete. Pivotal recommends that you run it independently of the SSH session, so that the process can continue running even if your connection to the jumpbox fails. The command above uses `nohup` but you could also run the command in a `screen` or `tmux` session.

If the commands completes successfully, continue to Step 9: Identify Your Deployment.

If the command fails, do the following:

1. Run restore-cleanup

   ```
   $ bbr director \
   --private-key-path PRIVATE-KEY \
   --username bbr \
   --host HOST \
   backup-cleanup
   ```
2. Ensure all the parameters in the command are set.
3. Ensure the BOSH Director credentials are valid.
4. Ensure the specified deployment exists.
5. Ensure the source deployment is compatible with the target deployment.
6. Ensure that the jumpbox can reach the BOSH Director.

Step 9: Identify Your Deployment

After logging in to your BOSH Director, run `bosh deployments` to identify the name of the BOSH deployment that contains PCF:

```
$ bosh -e DIRECTOR-IP --ca-cert PATH-TO-BOSH-SERVER-CERTIFICATE deployments

Name              Release(s)
cf-example        push-apps-manager-release/661.1.24
                  cf-backup-and-restore/0.0.1
                  binary-buildpack/1.0.11
                  capi/1.28.0
                  cf-autoscaling/91
                  cf-mysql/35
                  ...
```

In the above example, the name of the BOSH deployment that contains PCF is `cf-example`.

Step 10: Remove Stale Cloud IDs for All Deployments

For every deployment in the BOSH Director, run the following command:

```
$ bosh -e DIRECTOR-IP -d DEPLOYMENT-NAME -n cck \
  --resolution delete_disk_reference \
  --resolution delete_vm_reference
```

This reconciles the BOSH Director's internal state with the state in the IaaS. You can use the list of deployments returned in Step 9: Identify Your Deployment.

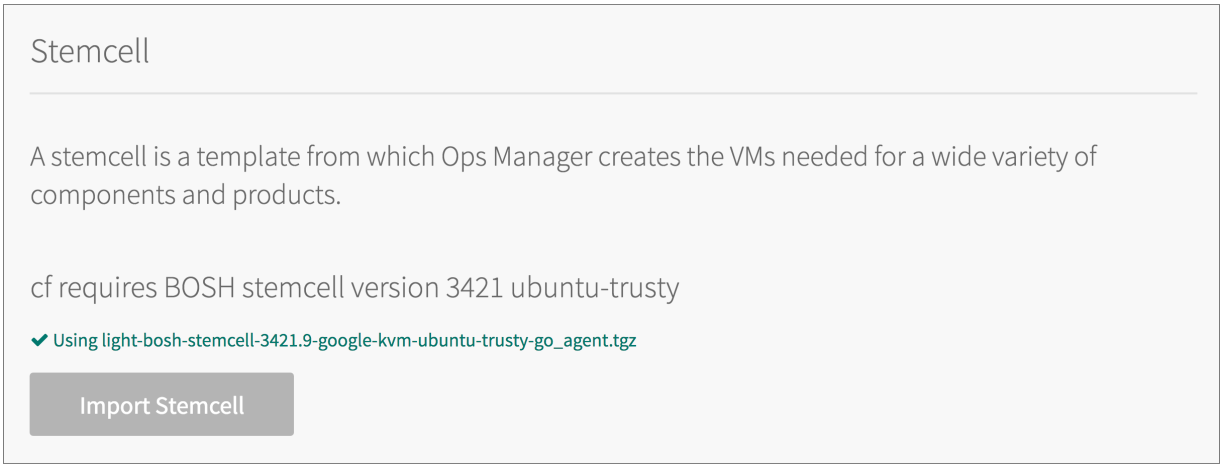
If the `bosh cloud-check` command does not successfully delete disk references and you see a message similar to the following, perform the additional procedures in the Remove Unused Disks section below.

```
Scanning 19 persistent disks: 19 OK, 0 missing ...
```

Step 11: Redeploy PAS

1. Perform the following steps to determine which stemcell is used by PAS:

    a. Navigate to the Ops Manager Installation Dashboard.
    b. Click the PAS tile.
    c. Click **Stemcell** and record the release number included in the displayed filename:

    In the screenshot above, the stemcell release number is **3421.9**.

    You can also retrieve the stemcell release using the BOSH CLI:

```
$ bosh -e DIRECTOR-IP deployments
Using environment '10.0.0.5' as user 'director' (bosh.*.read, openid, bosh.*.admin, bosh.read, bosh.admin)


Name                Release(s)                  Stemcell(s)                          Team(s)  Cloud Config
cf-9cb6995b7d746cd77438  push-apps-manager-release/661.1.24   bosh-google-kvm-ubuntu-trusty-go_agent/3421.9  -       latest
...
```

2. Download the stemcell from Pivotal Network.

3. Run the following command to upload the stemcell used by PAS:

```
$ bosh -e BOSH-DIRECTOR-IP \
  -d DEPLOYMENT-NAME \
  --ca-cert PATH-TO-BOSH-SERVER-CERTIFICATE \
  upload-stemcell \
  --fix PATH-TO-STEMCELL
```

4. If you have any other tiles installed, ensure you upload their stemcells if they are different from the PAS stemcell. Upload stemcells to the BOSH Director with `bosh upload-stemcell --fix PATH-TO-STEMCELL`, as in the command above.

5. From the Ops Manager Installation Dashboard, navigate to PAS **Resource Config**.

6. Ensure the number of instances for MySQL Server is set to `1`.

> **Warning**: Restore will fail if there is not exactly one MySQL Server instance deployed.

7. Ensure that all errands needed by your system are set to run.

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

> **Validation**: If your PAS uses an external blobstore, ensure that the PAS tile is configured to use a different blobstore before clicking **Apply Changes**. Otherwise it will attempt to connect to the blobstore that the existing PAS is using.

> **Validation**: Ensure your **System Domain** and **Apps Domain** under PAS **Domains** are updated to refer to the validation environment.

Step 12: Restore PAS

> **Validation:** If your apps must not be running after a restore run `bosh stop` on each `diego_cell` VM in the deployment.

1. If you use an external blobstore and copied it during the backup, restore the external blobstore with your IAAS specific tools before running PAS restore.

2. Run the BBR restore command from your jumpbox to restore PAS:

```
$ BOSH_CLIENT_SECRET=BOSH-PASSWORD \
  bbr deployment \
    --target BOSH-DIRECTOR-IP \
    --username BOSH-CLIENT \
    --deployment DEPLOYMENT-NAME \
    --ca-cert PATH-TO-BOSH-SERVER-CERTIFICATE \
    restore \
     --artifact-path PATH-TO-PAS-BACKUP
```

Replace the placeholder values as follows:

- `BOSH-CLIENT`, `BOSH-PASSWORD`: Use the BOSH UAA user provided in **Pivotal Ops Manager > Credentials > Uaa Bbr Client Credentials**.

  You can also retrieve the credentials using the Ops Manager API with a GET request to the following endpoint: `/api/v0/deployed/director/credentials/uaa_bbr_client_credentials`. For more information, see the Using the Ops Manager API topic.

- `BOSH-DIRECTOR-IP`: You retrieved this value in the Step 6: Retrieve BOSH Director Address and Credentials section.
- `DEPLOYMENT-NAME`: You retrieved this value in the Step 8: Identify Your Deployment section.
- `PATH-TO-BOSH-SERVER-CERTIFICATE`: This is the path to the BOSH Director's Certificate Authority (CA) certificate, if the certificate is not

verifiable by the local machine's certificate chain.
- `PATH-TO-PAS-BACKUP`: This is the path to the PAS backup you want to restore.

> **Validation**: If you ran `bosh stop` on each `diego_cell` before running `bbr restore`, you can now run `cf stop` on all apps and then run `bosh start` on each `diego_cell`. After this, all apps will be deployed in a stopped state.

3. If desired, scale the MySQL Server job back up to its previous number of instances by navigating to the **Resource Config** section of the PAS tile. After scaling the job, return to the Ops Manager Installation Dashboard and click **Apply Changes** to deploy.

(Optional) Step 13: Restore On-Demand Service Instances

> **Note**: These procedures restore the on-demand service instances but do not restore service instance data.

If you have on-demand service instances provisioned by an on-demand service broker, perform the following steps to restore them after successfully restoring PCF:

1. Use the Cloud Foundry Command Line Interface (cf CLI) to target your PCF deployment:

```
$ cf api api.YOUR-SYSTEM-DOMAIN
```

2. Log in:

```
$ cf login
```

3. Perform the following steps to make a list of all the service instances provisioned by your on-demand service broker:

   a. List your service offerings:

```
$ cf curl /v2/services
```

   b. Record the GUID of the on-demand service offering you want to restore by examining the value for `guid` under `metadata`:

```
"metadata": {
"guid": "ab2b01cc-2a22-525a-a333-e6e666a6aa66",
"url": "/v2/services/ab2b01cc-2a22-525a-a333-e6e666a6aa66",
"created_at": "2017-02-10T18:19:35Z",
"updated_at": "2017-02-10T18:19:35Z"
```

   c. List all service plans for the service offering, replacing `SERVICE-OFFERING-GUID` with the GUID obtained in the previous step:

```
$ cf curl /v2/services/SERVICE-OFFERING-GUID/service_plans
```

   d. Record the GUID of each service plan by examining the value for `guid` under `metadata`.
   e. For each service plan, list all service instances:

```
$ cf curl /v2/service_plans/SERVICE-PLAN-GUID/service_instances
```

   f. Record the GUID of each service instance by examining the value for `guid` under `metadata`.

4. Perform the following steps to obtain the BOSH credentials used by your on-demand service broker:

   a. Navigate to `https://YOUR-OPS-MAN-FQDN/api/v0/staged/products` in a browser to obtain the product GUID of your tile.
   b. Navigate to `https://YOUR-OPS-MAN-FQDN/api/v0/staged/products/PRODUCT-GUID/manifest` to obtain your product's staged manifest.
   c. Copy the manifest into a file on your local machine called `manifest.json`.
   d. Run the following command to find the name of the deployment's on-demand broker instance group:

```
$ cat manifest.json | jq '(.instance_groups[].name )' | grep on-demand-broker | grep -v -E "register|smoke"
> redis-on-demand-broker
```

   e. Run the following command to extract the BOSH credentials:

```
$ cat manifest.json | jq '(.instance_groups[] |
select(.name == "redis-on-demand-broker").jobs[] |
select(.name == "broker").properties.bosh.authentication.uaa )'
```

5. SSH into your Ops Manager VM. For more information, see the SSH into Ops Manager section of the *Advanced Troubleshooting with the BOSH CLI* topic.

6. Using the BOSH credentials retrieved above, authenticate with your BOSH Director by running the following commands with the BOSH CLI v2:

```
$ export BOSH-CLIENT=YOUR-CLIENT-ID
$ export BOSH-CLIENT-SECRET=YOUR-CLIENT-SECRET
$ bosh alias-env director -e DIRECTOR-IP \
--ca-cert /var/tempest/workspaces/default/root_ca_certificate
```

7. Using the list of service instance GUIDs gathered above, deploy each instance with the following commands:

```
$ bosh -e director manifest \
-d service-instance_SERVICE-INSTANCE-GUID > /tmp/manifest.yml
$ bosh -e director \
-d service-instance_SERVICE-INSTANCE-GUID deploy /tmp/manifest.yml
```

8. After deploying all service instances, remove the manifest from `tmp`.

```
$ rm /tmp/manifest.yml
```

9. Any PAS apps bound to these services will have to be restarted to pick up the recreated service instances.

Remove Unused Disks

If `bosh cloud-check` does not clean up all disk references, you must manually delete the disks from a previous deployment that will prevent recreated deployments from working.

Warning: This is a very destructive operation.

To delete the disks, perform one of the following procedures:

- Use the BOSH CLI to delete the disks by performing the following steps:
  1. Target the redeployed BOSH Director using the BOSH CLI by performing the procedures in Step 6: Retrieve BOSH Director Address and Credentials.
  2. List the deployments by running the following command:

     ```
     $ bosh -e DIRECTOR-IP --ca-cert /var/tempest/workspaces/default/root_ca_certificate deployments
     ```

  3. Delete each deployment with the following command:

     ```
     $ bosh -d DEPLOYMENT-NAME delete-deployment
     ```

- Log in to your IaaS account and delete the disks manually. Run the following command to retrieve a list of disk IDs:

  ```
  $ bosh -e DIRECTOR-IP --ca-cert /var/tempest/workspaces/default/root_ca_certificate instances
  ```

Once the disks are deleted, continue with Step 9: Remove Stale Cloud IDs for All Deployments.

Setting Up Your Jumpbox for BBR

Page last updated:

This topic describes how to set up your jumpbox for BOSH Backup and Restore (BBR).

To use BBR to back up and restore your Pivotal Cloud Foundry (PCF) deployment, you must first set up a jumpbox to run BBR from.

Step 1: Set Up Your Jumpbox

Set up your jumpbox with the following settings:

- The jumpbox must have sufficient space for the backup. A PCF backup will be at least 1.5 GB in size.
- BBR connects to the VMs at their private IP address, so the jumpbox needs to be in the same network as the deployed VMs. BBR does not support SSH gateways.
- BBR copies the backed-up data from the VMs to the jumpbox, so ensure you have minimal network latency between them to reduce transfer times.

Consult the following table for more information about the network access permissions required by BBR.

VM	Default Port	Description
BOSH Director	25555	BBR interacts with the BOSH Director API.
Deployed Instances	22	BBR uses SSH to orchestrate the backup on the instances.
BOSH Director UAA	8443	BBR interacts with the UAA API for authentication, if necessary.

Step 2: Transfer BBR to Your Jumpbox

Perform the following steps to transfer BBR to your jumpbox:

1. Download the latest BBR release.
2. Change the permissions of `bbr` in order to make it executable:

```
$ chmod a+x bbr
```

3. SCP the binary to your jumpbox:

```
$ scp LOCAL_PATH_TO_BBR/bbr JUMPBOX_USER/JUMPBOX_ADDRESS
```

If your jumpbox has access to the internet, you can also SSH into your jumpbox and use `wget`:

```
$ ssh JUMPBOX_USER/JUMPBOX_ADDRESS -i YOUR_CERTIFICATE.pem
$ wget BBR_RELEASE_URL
$ chmod a+x bbr
```

Restoring a PAS Backup In-Place

Page last updated:

This document describes how to run an *in-place* restore of an Pivotal Application Service (PAS) backup created with BOSH Backup and Restore (BBR). An in-place restore differs from a normal restore in that you do not tear down and redeploy your environment before the restore. Instead, you restore the PAS backup to the same PCF environment that the backup was created from.

> **Warning**: An in-place restore erases any changes made to PAS since the backup was created. Follow the procedures in this topic only for sandbox or development environments, or when recovering from a disaster that affected PAS data.

> **Warning**: For PCF v2.0.0, BBR only supports backup and restore of environments with zero or one CredHub instances.

Step 1: Scale Down MySQL

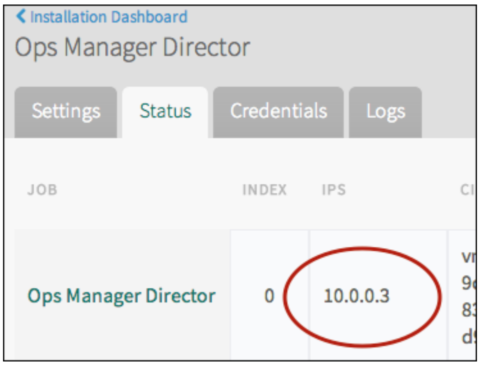
1. From the Ops Manager Installation Dashboard, navigate to PAS **Resource Config**.
2. Ensure the number of instances for MySQL Server is set to `1`.

   > **Warning**: Restore fails if there is not exactly one MySQL Server instance deployed.

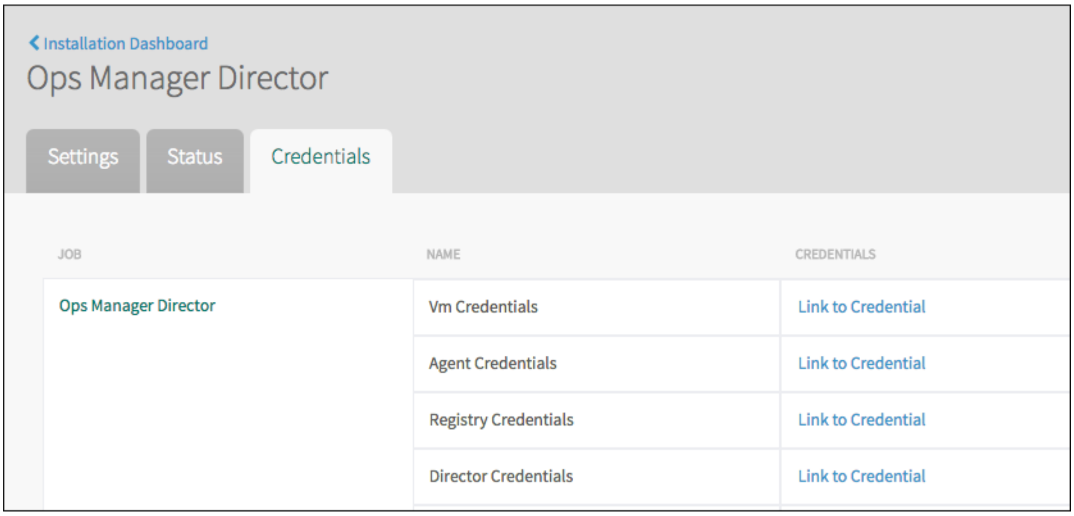
3. Return to the Ops Manager Installation Dashboard and click **Apply Changes** to redeploy.

Step 2: Retrieve BOSH Director Address and Credentials

Perform the following steps to retrieve the IP address of your BOSH Director and the credentials for logging in from the Ops Manager Director tile:

1. Install the BOSH v2 CLI on a machine outside of your PCF deployment. You can use the jumpbox for this task.
2. From the Installation Dashboard in Ops Manager, select **Ops Manager Director > Status** and record the IP address listed for the Director. You access the BOSH Director using this IP address. 

3. Click **Credentials** and record the Director credentials.



4. From the command line, log into the BOSH Director using the IP address and credentials that you recorded:

```
$ bosh -e DIRECTOR_IP \
--ca-cert PATH-TO-BOSH-SERVER-CERT log-in
Email (): director
Password (): ******************
Successfully authenticated with UAA
Succeeded
```

Step 3: Identify Your Deployment

After logging in to your BOSH Director, run `bosh deployments` to identify the name of the BOSH deployment that contains PCF:

```
$ bosh -e DIRECTOR_IP --ca-cert /var/tempest/workspaces/default/root_ca_certificate deployments

Name            Release(s)
cf-example        push-apps-manager-release/661.1.24
              cf-backup-and-restore/0.0.1
              binary-buildpack/1.0.11
              capi/1.28.0
              cf-autoscaling/91
              cf-mysql/35
              ...
```

In the above example, the name of the BOSH deployment that contains PCF is `cf-example`.

Step 4: Restore PAS

Note: If your apps must not run after a restore, run `bosh stop` on each `diego_cell` VM in the deployment.

1. If you use an external blobstore and copied it during the backup, restore the external blobstore with your IaaS-specific tools before restoring PAS.

2. Run the BBR restore command from your jumpbox to restore PAS:

```
$ BOSH_CLIENT_SECRET=BOSH-PASSWORD \
 bbr deployment \
  --target BOSH-DIRECTOR-IP \
  --username BOSH-CLIENT \
  --deployment DEPLOYMENT-NAME \
  --ca-cert PATH-TO-BOSH-SERVER-CERT \
  restore \
   --artifact-path PATH-TO-ERT-BACKUP
```

Replace the placeholder values as follows:

- `BOSH_CLIENT`, `BOSH_PASSWORD`: Use the BOSH UAA user provided in **Pivotal Ops Manager > Credentials > Uaa Bbr Client Credentials**.

  You can also retrieve the credentials using the Ops Manager API with a GET request to the following endpoint: `/api/v0/deployed/director/credentials/uaa_bbr_client_credentials`. For more information, see the Using the Ops Manager API topic.
- `BOSH_DIRECTOR_IP`: You retrieved this value in the Step 2: Retrieve BOSH Director Address and Credentials section.
- `DEPLOYMENT-NAME`: You retrieved this value in the Step 3: Identify Your Deployment section.
- `PATH_TO_BOSH_SERVER_CERT`: The path to the BOSH Director's Certificate Authority (CA) certificate, if the certificate is not verifiable by the local machine's certificate chain.
- `PATH_TO_ERT_BACKUP`: The path to the PAS backup you want to restore.

> **Validation:** If you ran `bosh stop` on each `diego_cell` before running `bbr restore`, you can now run `cf stop` on all apps and then run `bosh start` on each `diego_cell`. After this, all apps will be deployed in a stopped state.

3. Perform the following steps after restoring PAS:

   a. Retrieve the MySQL admin credentials from CredHub using the Ops Manager API:

      i. Perform the procedures in the Using the Ops Manager API topic to authenticate and access the Ops Manager API.

      ii. Use the `GET /api/v0/deployed/products` endpoint to retrieve a list of deployed products, replacing `UAA-ACCESS-TOKEN` with the access token recorded in the previous step:

```
$ curl "https://OPS-MAN-FQDN/api/v0/deployed/products" \
    -X GET \
    -H "Authorization: Bearer UAA-ACCESS-TOKEN"
```

      iii. In the response to the above request, locate the product with an `installation_name` starting with `cf-` and copy its `guid`.

      iv. Run the following `curl` command, replacing `PRODUCT-GUID` with the value of `guid` from the previous step:

```
$ curl "https://OPS-MAN-FQDN/api/v0/deployed/products/PRODUCT-GUID/variables?name=mysql-admin-credentials" \
    -X GET \
    -H "Authorization: Bearer UAA-ACCESS-TOKEN"
```

      v. Record the MySQL admin credentials from the response to the above request.

   b. List the VMs in your deployment:

```
$ bosh -e DIRECTOR-IP --ca-cert /var/tempest/workspaces/default/root_ca_certificate \
-d DEPLOYMENT-NAME \
ssh
```

   c. Select the `mysql` VM to SSH into.

   d. From the `mysql` VM, run the following command:

```
$ sudo /var/vcap/packages/mariadb/bin/mysql -u root -p
```

   When prompted, enter the MySQL admin password.

   e. At the MySQL prompt, run the following command:

```
mysql> use silk; drop table subnets; drop table gorp_migrations;
```

   f. Exit MySQL:

```
mysql> exit
```

   g. Exit the `mysql` VM:

```
$ exit
```

   h. List the VMs in your deployment:

```
$ bosh -e DIRECTOR-IP --ca-cert /var/tempest/workspaces/default/root_ca_certificate \
-d DEPLOYMENT-NAME \
ssh
```

   i. SSH onto each `diego_database` VM and run the following command:

```
$ sudo monit restart silk-controller
```

Restored apps will begin to start. The amount of time it takes for all apps to start depends on the number of app instances, the resources available to the underlying infrastructure, and the value of the **Max Inflight Container Starts** field in the PAS tile.

4. If desired, scale the MySQL Server job back up to its previous number of instances by navigating to the **Resource Config** section of the PAS tile. After scaling the job, return to the Ops Manager Installation Dashboard and click **Apply Changes** to deploy.

Troubleshooting BBR

Page last updated:

This topic lists common troubleshooting scenarios and their solutions when using BOSH Backup and Restore (BBR) to back up and restore Pivotal Cloud Foundry (PCF).

Troubleshooting During a Restore

Symptom

The restore fails with a MySQL monit start timeout.

While running the BBR restore command, restoring the job `mysql-restore` fails with the following error:

```
1 error occurred:

* restore script for job mysql-restore failed on mysql/0.
...
Monit start failed: Timed out waiting for monit: 2m0s
```

Explanation

This happens when `mariadb` fails to start within the timeout period. It will end up in an "Execution Failed" state and `monit` will never try to start it again.

Solution

Ensure that your MySQL Server cluster has only one instance. If there are more than one instances of MySQL Server, the restore will fail with a `monit start` timeout. Scale down to one instance and retry.

If your MySQL Server cluster is already scaled down to one node, it may have taken longer than normal to restart the cluster. Follow the procedure below to manually verify and retry.

1. List the VMs in your deployment:

   ```
   $ bosh -e DIRECTOR_IP --ca-cert /var/tempest/workspaces/default/root_ca_certificate \
   -d DEPLOYMENT_NAME \
   ssh
   ```

2. Select the `mysql` VM to SSH into.

3. From the `mysql` VM, run the following command to check that the `mariadb` process is running:

   ```
   $ ps aux | grep mariadb
   ```

4. Run the following command to check that `monit` reports `mariadb_ctrl` is in an "Execution Failed" state:

   ```
   $ sudo monit summary
   ```

5. If so, run the following command from the `mysql` VM to disable monitoring:

   ```
   $ monit unmonitor
   ```

6. Run the following command to enable monitoring:

   ```
   $ monit monitor
   ```

7. After a few minutes, run the following command:

```
$ monit summary
```

The command should report that all the processes are running.

8. Re-attempt the restore with BBR.

Symptom

The deployment does not match the structure of the backup.

The following error displays:

```
Deployment 'deployment-name' does not match the structure of the provided backup
```

Explanation

The instance groups with the restore scripts in the destination environment don't match the backup metadata. For example, they may have the wrong number of instances of a particular instance group, or the metadata names an instance group that doesn't exist in the destination environment.

Solution

BBR only supports restoring to an environment that matches your original environment. Pivotal recommends altering the destination environment to match the structure of the backup.

General Troubleshooting

Symptom

BBR displays an error message containing "SSH Dial Error" or another connection error.

Explanation

The jumpbox and the VMs in the deployment are experiencing connection problems.

Solution

Perform the following steps:

1. Ensure your deployment is healthy by running `bosh vms`.
2. Run `bbr deployment backup-cleanup` in order to clean up the data from the failed backup on the instances. Otherwise, further BBR commands will fail.
3. Repeat the BBR operation.

Symptom

BBR backup or restore fails with a metadata error:

```
1 error occurred:
error 1:
An error occurred while running metadata script for job redis-server on redis/0ce9f81f-1756-480b-8e3e-a4609b14b6a6: error from metadata
```

Explanation

There is a problem with your PCF install.

Solution

Contact Pivotal Support

BOSH Backup and Restore

This guide documents BOSH Backup and Restore (BBR), a framework for backing up and restoring BOSH deployments and BOSH Directors.

Overview

BBR orchestrates triggering the backup or restore process on the BOSH deployment or BOSH Director, and transfers the backup artifacts to and from the BOSH deployment or BOSH Director.

For more information about installing and using BBR, see the Installing BOSH Backup and Restore, Backing Up with BOSH Backup and Restore, and Restoring with BOSH Backup and Restore topics. Information for release authors and developers is also available in the BOSH Backup and Restore Developer's Guide.

Supported Components

BBR is a binary that can back up and restore BOSH deployments and BOSH Directors. BBR requires that the backup targets supply scripts that implement the backup and restore functions.

BBR is not dependent on a particular version of BOSH. However, a BOSH deployment must have its backup and restore scripts packaged in the releases to be backed up and restored with BBR. For more information, consult the documentation for the deployment.

You can back up and restore the following BOSH deployments with BBR:

- BOSH Director, including UAA and CredHub

BBR supports BOSH Directors which use one of the following:

- Basic Auth
- Client/Client-Secret UAA Authentication

Contract

BBR sets out a contract with BOSH release authors to call designated backup and restore scripts in a specific order.

This approach has the following advantages:

- The deployment itself encapsulates the knowledge of how to back up and restore the deployment.
- Because responsibility for writing and maintaining scripts sits with the release author, scripts can change as the deployment changes and do not get out of sync.

Backup Scripts

1. **pre-backup-lock**: The pre-backup lock script locks the job so backups are consistent across the cluster.
2. **backup**: The backup script backs up the release.
3. **post-backup-unlock**: The post-backup unlock script unlocks the job after the backup is complete.

Restore Scripts

1. **pre-restore-lock**: The pre-restore-lock script locks the job so the restore is consistent across the cluster.
2. **restore**: The restore script restores the release.
3. **post-restore-unlock**: The post-restore-unlock script unlocks the job after the restore is complete.

Workflow

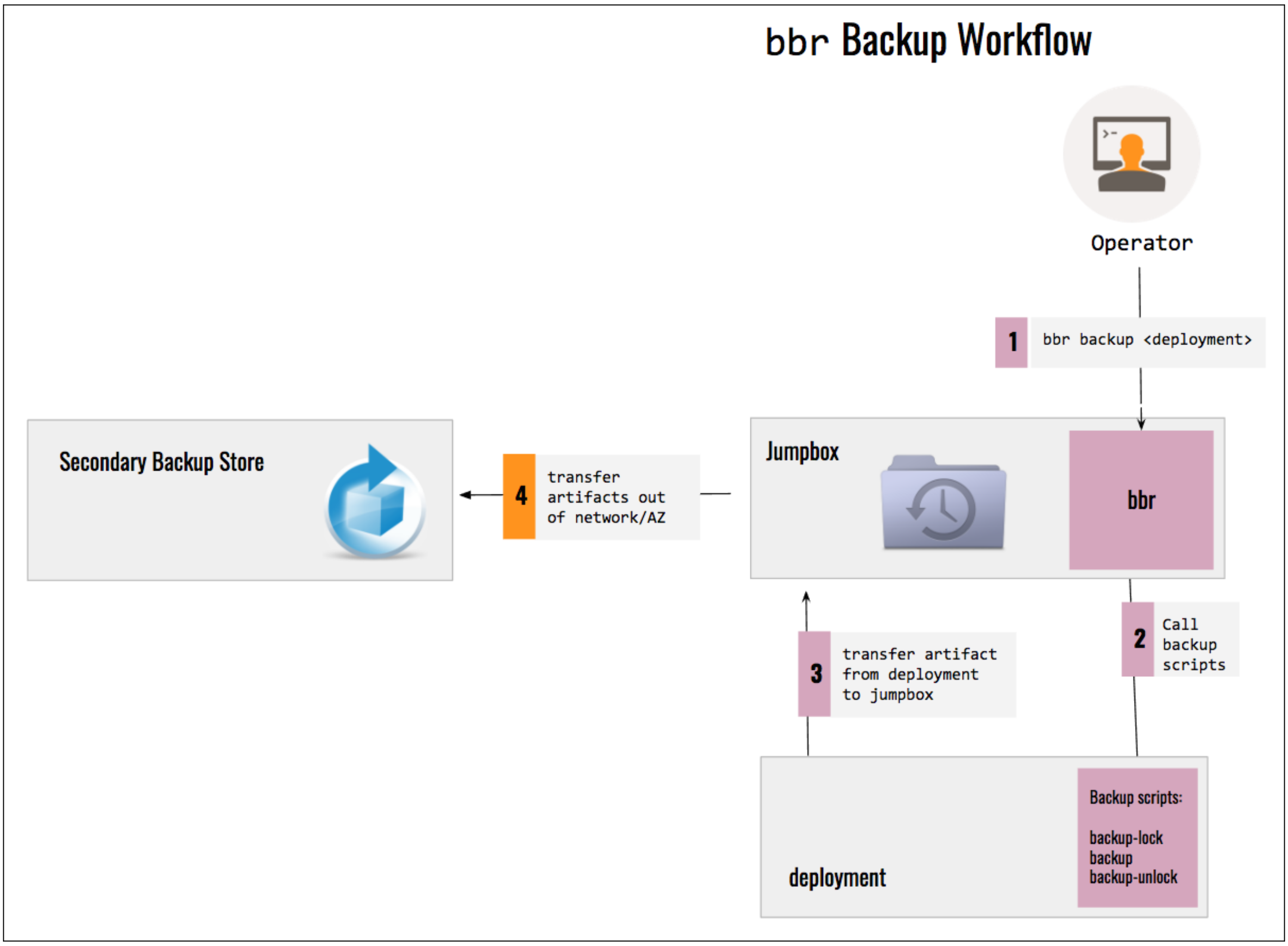
Operators download the BBR binary and transfer it to a jumpbox. Then they run BBR from the jumpbox, specifying the name of the BOSH deployment or BOSH Director to back up.

BBR examines the jobs in the BOSH deployment or BOSH Director, and triggers the pre-backup lock, backup, and post-backup unlock scripts.

Scripts in the same stage are all triggered together. For instance, BBR triggers all pre-backup lock scripts before any backup scripts. Scripts within a stage may be triggered in any order.

The backup artifacts are drained to the jumpbox where the operator can transfer them to storage and use them to restore the BOSH deployment or BOSH Director.

The following diagram shows a sample backup flow.



Syntax

This section provides syntax information for the BBR binary. For detailed procedures on how to use BBR to back up and restore a BOSH Director or BOSH deployment, see the Backing Up with BOSH Backup and Restore and Restoring with BOSH Backup and Restore topics.

The syntax for the BBR binary is:

```
bbr [command] [arguments...] [subcommand]
```

The options for `[command]` are:

- `deployment`: Specifies that the target of BBR is a BOSH deployment
- `director`: Specifies that the target of BBR is a BOSH Director
- `help`: Prints help text

- `version` : Prints the version of the `bbr` binary

The `[arguments]` are specific to the command.

BOSH Director

The arguments to specify when running BBR for a BOSH Director are:

```
$ bbr director \
 --debug (OPTIONAL) \
 --private-key-path PATH_TO_PRIVATE_KEY \
 --username USER_NAME \
 --host HOST \
 SUB-COMMAND [--artifact-path PATH_TO_ARTIFACT_TO_RESTORE]
```

The parameters are:

- `--debug` : This is an optional flag to display debug output.
- `--private-key-path` : This is the path to the SSH private key used to connect to the BOSH Director.
- `--username` : This is the SSH username of the BOSH Director.
- `--host` : This is the address of the BOSH Director with an optional port that defaults to 22. If the BOSH Director is public, this is a URL, such as `https://my-bosh.xxx.cf-app.com` . Otherwise, this is the BOSH Director IP address.
- `--artifact-path` : This is the path to the BOSH Director backup you want to restore.

BOSH Deployment

The arguments to specify when running BBR for a BOSH deployment are:

```
$ BOSH_CLIENT_SECRET=BOSH_CLIENT_SECRET \
 bbr deployment \
 --debug (OPTIONAL)
 --target BOSH_TARGET \
 --username BOSH_CLIENT \
 --deployment DEPLOYMENT_NAME \
 --ca-cert PATH_TO_BOSH_SERVER_CERTIFICATE \
  SUB-COMMAND [--artifact-path PATH_TO_ARTIFACT_TO_RESTORE]
```

The arguments are:

- `--debug` : This is an optional flag to display debug output.
- `BOSH_CLIENT` , `BOSH_CLIENT_SECRET` : If you have a BOSH Director with User Account and Authentication (UAA) as the authentication provider, use a UAA client as the username and a UAA client secret as the password. If you have a BOSH Director with basic auth configured, use your username and password.
- `--target` : This is the FQDN or IP address of your BOSH Director.
- `--deployment` : This is the name of the deployment you want to back up. For a list of deployments, run `bosh deployments` .
- `--ca-cert` : This is the path to the BOSH Director's Certificate Authority (CA) certificate if the certificate is not verifiable by the local machine's certificate chain.
- `--artifact-path` : This is the path to the BOSH deployment backup you want to restore.

Subcommands

BBR supports five subcommands:

- `backup`
- `restore`
- `pre-backup-check`
- `backup-cleanup`
- `restore-cleanup`

Using Ops Manager

Ops Manager is a web application that you use to deploy and manage a Pivotal Cloud Foundry (PCF) PaaS. This is a guide to deploying and using Ops Manager.

Browser Support

Ops Manager is compatible with current and recent versions of all major browsers. Pivotal recommends using the current version of Chrome, Firefox, or Safari for the best Ops Manager experience.

Ops Manager API

Use the Ops Manager API to automate any Ops Manager task.

See the Using Ops Manager API topic to learn how to get started using the Ops Manager API. To view the Ops Manager API documentation, browse to `https://YOUR-OPS-MANAGER-FQDN/docs`.

Using Ops Manager and Installed Products

- Understanding the Ops Manager Interface
- Adding and Deleting Products
- Applying Changes to Ops Manager Director
- Retrieving Credentials from Your Deployment
- Understanding Floating Stemcells
- Configuring Ops Manager Director for VMware vSphere
- vSphere Service Account Requirements
- Creating UAA Clients for BOSH Director
- Configuring Ops Manager Director for OpenStack
- Configuring PAS for vSphere
- Installing PAS after Deploying Pivotal Cloud Foundry on OpenStack
- Using Your Own Load Balancer
- Understanding Pivotal Cloud Foundry User Types
- Creating and Managing Ops Manager User Accounts
- Creating New PAS User Accounts
- Logging in to Apps Manager
- Adding Existing SAML or LDAP Users to a Pivotal Cloud Foundry Deployment
- Deleting an AWS Installation from the Console
- Modifying Your Ops Manager Installation and Product Template Files
- Managing Errands in Ops Manager

Backing Up

- Backing Up and Restoring Pivotal Cloud Foundry

Monitoring, Logging, and Troubleshooting

- Monitoring Virtual Machines in Pivotal Cloud Foundry
- Diagnosing Problems in PCF

- Troubleshooting Problems in PCF
- Troubleshooting Ops Manager for VMware vSphere
- Recovering MySQL from PAS Downtime
- Advanced Troubleshooting with the BOSH CLI

Pivotal

Using the Ops Manager API

This topic explains how to start using the Ops Manager API.

Platform operators use the Ops Manager API to automate deployments, retrieve and manage credentials, and otherwise work under the hood of the Ops Manager interface.

Tile developers use the Ops Manager API to test and debug Pivotal Cloud Foundry (PCF) product tiles.

For the complete Ops Manager API documentation, browse to `https://YOUR-OPS-MANAGER-FQDN/docs`, adding `/docs` to the URL of your Ops Manager.

Requirements

You must install the User Account and Authentication Command Line Interface (UAAC) to perform the procedures in this topic. To install the UAAC, run the following command from a terminal window:

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

Step 1: Authenticate

To use the Ops Manager API, you must authenticate and retrieve a token from the Ops Manager User Account and Authentication (UAA) server. For more information about UAA, see the User Account and Authentication (UAA) Server topic.

Perform the procedures in the Internal Authentication or External Identity Provider section below depending on which authentication system you configured for Ops Manager.

Internal Authentication

If you configured your Ops Manager for Internal Authentication, perform the following procedures specific to your IaaS:

vSphere

You need the credentials used to import the PCF .ova or .ovf file into your virtualization system.

1. From a command line, run `ssh ubuntu@OPS-MANAGER-FQDN` to SSH into the Ops Manager VM. Replace `OPS-MANAGER-FQDN` with the fully qualified domain name of Ops Manager.
2. When prompted, enter the password that you set during the .ova deployment into vCenter. For example:

   ```
   $ ssh ubuntu@my-opsmanager-fqdn.example.com
   Password: ***********
   ```

3. Proceed to Authenticate into Ops Manager.

AWS, Azure, and OpenStack

1. Locate the Ops Manager FQDN on the AWS **EC2 instances** page, Azure **Virtual machines** page, or the OpenStack **Access & Security** page.
2. Run `chmod 600 ops_mgr.pem` to change the permissions on the `.pem` file to be more restrictive:

   ```
   $ chmod 600 ops_mgr.pem
   ```

3. Run `ssh -i ops_mgr.pem ubuntu@OPS-MANAGER-FQDN` to SSH into the Ops Manager VM. Replace `OPS-MANAGER-FQDN` with the fully qualified domain name of Ops Manager. For example:

   ```
   $ ssh -i ops_mgr.pem ubuntu@my-opsmanager-fqdn.example.com
   ```

4. Proceed to Authenticate into Ops Manager.

GCP

1. Confirm that you have installed the gcloud CLI. If you do not have the gcloud CLI, see the Google Cloud Platform documentation.
2. Run `gcloud config set project MY-PROJECT` to configure your Google Cloud Platform project. For example:

```
$ gcloud config set project gcp
```

3. Run `gcloud auth login MY-GCP-ACCOUNT`. For example:

```
$ gcloud auth login user@example.com
```

4. Run `gcloud compute ssh MY-INSTANCE --zone MY-ZONE`. For example:

```
$ gcloud compute ssh om-pcf-1a --zone us-central1-b
```

5. Run `sudo su - ubuntu` to switch to the `ubuntu` user.
6. Proceed to Authenticate into Ops Manager.

Authenticate into Ops Manager

1. After successfully SSHing into the Ops Manager VM, use the UAAC to target your Ops Manager UAA server:

```
$ uaac target https://OPS-MAN-FQDN/uaa
```

2. Retrieve your token to authenticate:

```
$ uaac token owner get
Client ID: opsman
Client secret: [Leave Blank]
User name: OPS-MAN-USERNAME
Password: OPS-MAN-PASSWORD
```

Replace `OPS-MAN-USERNAME` and `OPS-MAN-PASSWORD` with the credentials that you use to log in to the Ops Manager web interface.

External Identity Provider

If you configured your Ops Manager for an external Identity Provider with SAML, perform the following steps:

1. From your local machine, target your Ops Manager UAA server:

```
$ uaac target https://OPS-MAN-FQDN/uaa
```

2. Retrieve your token to authenticate. When prompted for a passcode, retrieve it from `https://OPS-MAN-FQDN/uaa/passcode`.

```
$ uaac token sso get
Client ID: opsman
Client secret: [Leave Blank]
Passcode: YOUR-PASSCODE
```

If authentication is successful, the UAAC displays the following message: `Successfully fetched token via owner password grant.`

Step 2: Access the API

Ops Manager uses authorization tokens to allow access to the API. You must pass an access token to the API endpoint in a header that follows the format `Authorization: Bearer YOUR-ACCESS-TOKEN`.

The following example procedure retrieves a list of deployed products. See the Ops Manager API documentation at `https://YOUR-OPS-MANAGER-FQDN/docs` for the full range of API endpoints.

If you use Internal Authentication, you must perform the following procedures from the Ops Manager VM. If you use an External Identity Provider, you may perform the procedures from your local machine.

1. List your tokens:

   ```
   $ uaac contexts
   ```

   Locate the entry for your Ops Manager FQDN. Under `client_id: opsman`, record the value for `access_token`.

2. Use the `GET /api/v0/deployed/products` endpoint to retrieve a list of deployed products, replacing `UAA-ACCESS-TOKEN` with the access token recorded in the previous step:

   ```
   $ curl "https://OPS-MAN-FQDN/api/v0/deployed/products" \
     -X GET \
     -H "Authorization: Bearer UAA-ACCESS-TOKEN"
   ```

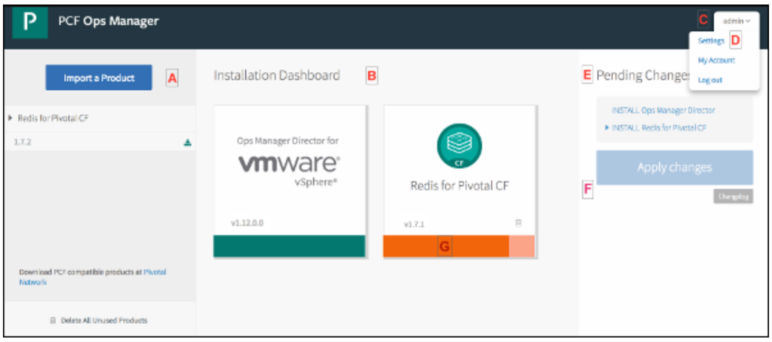
   The request produces the following response:


   ```
   [{"installation_name":"p-bosh","guid":"p-bosh
   -00000000000000000000","type":"p-
   bosh","product_version":"1.10.0.
   0"},{"installation_name":"cf-
   00000000000000000000","guid":"cf-0000000000000
   0000000","type":"cf","product_version":"1.10.0"}]
   ```


Understanding the Ops Manager Interface

Page last updated:

This topic describes key features of the Pivotal Cloud Foundry (PCF) Operations Manager interface.

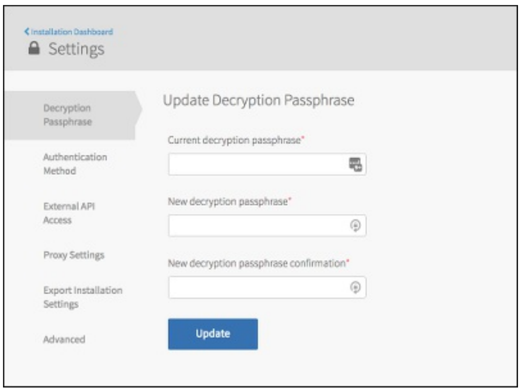


- **A**— Displays a list of products you have imported that are ready for installation.
  - Click the **Import a Product** link to add a new product to Ops Manager.
  - If an upgrade is available, an active **Upgrade** button appears when you hover over the name of the product. If you are using the Pivotal Network API, the latest version of an existing product appears automatically.
  - Click **Delete All Unused Products** to delete any unused products.
- **B**— **Installation Dashboard**: Displays a product tile for each installed product.
- **C**— **User account menu**: Use this menu to navigate to your **Settings** page, view **My Account** to change your email and password, or log out of the **Installation Dashboard**.
- **D**— **Settings**: This menu option opens a page with several configuration panes. See the Settings Page section of this topic for details.
- **E**— **Pending Changes view**: Displays queued products and updates that will install during the next deploy. Clicking on a product expands its list of errands and lets you change the errand run rules for the next deploy. For more information, see Managing Errands in Ops Manager.
- **F**— **Apply Changes button**: Clicking the button applies pending changes, as listed, to your deployment. You can also view the logs of past installation updates by clicking **Change Log**.
- **G**— **Orange colored bar**: Indicates that additional configuration for the product tile is required before deployment. Click on the product tile to complete its configuration. In addition, the **Apply Changes** button is low lit to indicate that changes cannot be applied without additional product configuration.

> **Note**: When an update depends on prerequisites, the prerequisites automatically install first.

Settings Page

Navigate to the **Settings** page by clicking on your user name located at the upper right corner of the screen and selecting **Settings**.



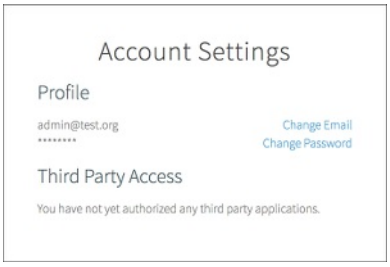
The following list describes each configuration pane:

- **Decryption Passphrase**: Reset your **decryption passphrase**.

- **Authentication Method**: You can switch Identity Providers by entering your **Decryption passphrase**, **Saml idp metadata**, and optionally, your **BOSH IdP metadata**. For more information about setting up your Identity Provider, view the following instructions for your configuration:
    - Amazon Web Services
    - Google Cloud Platform
    - Microsoft Azure
    - OpenStack
    - vSphere
- **External API Access**: Enter your Pivotal Network API token to connect your **Installation Dashboard** to the Pivotal Network.
- **Proxy Settings**: If you are using a proxy to connect to Ops Manager, update your **Proxy Settings** by providing a **HTTP proxy**, **HTTPS proxy**, or **No proxy**.
- **Export Installation Settings**: Exports the current installation with all of its assets. When you export an installation, the exported file contains references to the installation IP addresses. It also contains the base VM images and necessary packages. As a result, an export can be very large (as much as 5 GB or more).
- **Advanced**:
    - **Download activity data** - Downloads a directory containing the config file for the installation, the deployment history, and version information.
    - **Download Root CA Cert** - Use this to download the root CA certificate of your deployment as an alternative to curling the Ops Manager API.
    - **View diagnostic report** - Displays various types of information about the configuration of your deployment.
    - **Delete this Installation**

My Account (Account Settings) page

To change your email and password, navigate to the **My Account** page by clicking on your user name located at the upper right corner of the screen and selecting **My Account**.



Adding and Deleting Products

Page last updated:

Refer to this topic for help adding and deleting additional products from your Pivotal Cloud Foundry (PCF) installation, such as Pivotal RabbitMQ® for PCF.

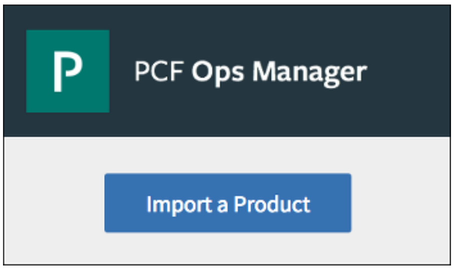
Note: In Ops Manager 2.0, all product tiles use floating stemcells by default. This increases the security of your deployment by enabling tiles to automatically use the latest patched version of a stemcell, but it may significantly increase the amount of time required by a tile upgrade. Review the Understanding Floating Stemcells topic for more information.

Adding and Importing Products

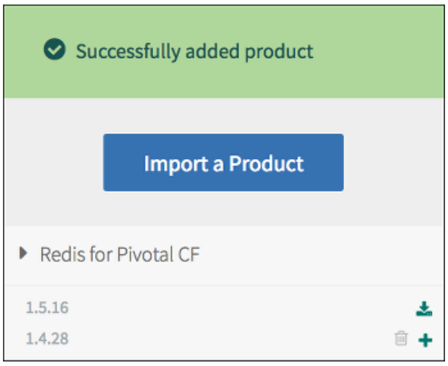
1. Download PCF-compatible products at Pivotal Network.

2. Navigate to your Ops Manager **Installation Dashboard** and log in.

3. Click **Import a Product**.

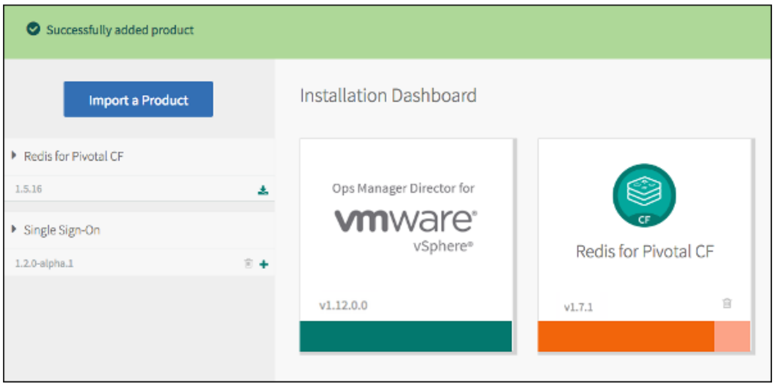


4. Select the .pivotal file that you downloaded from Pivotal Network or received from your software distributor, then click **Open**. If the product is successfully added, it appears in the your product list. If the product you selected is not the latest version, the most up-to-date version will appear on your product list.



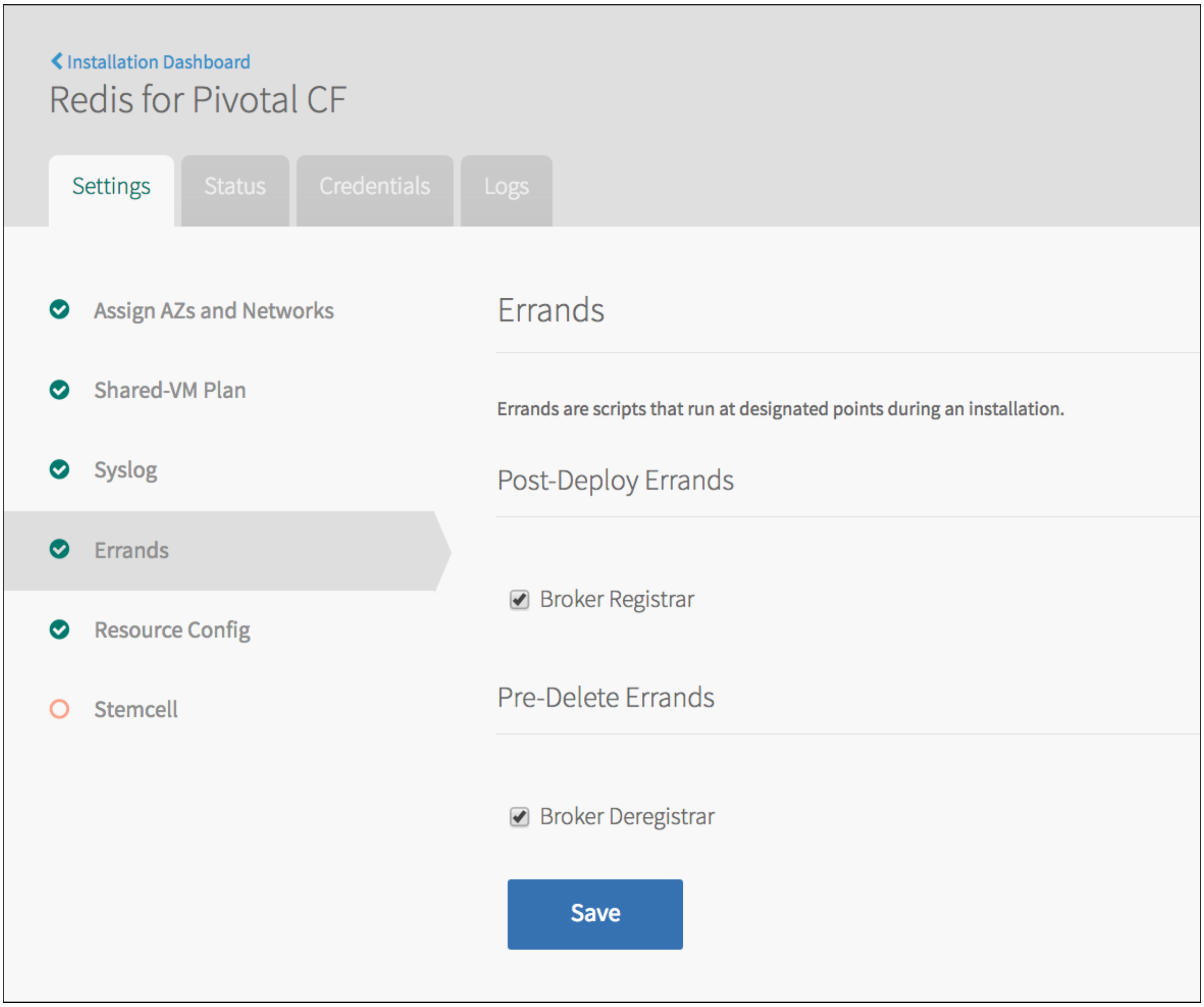
5. Add the product tile to the **Installation Dashboard** by clicking the green plus sign icon.

6. The product tile appears in the Installation Dashboard. If the product requires configuration, the tile appears orange. If necessary, configure the product.



7. **(Optional)** In the product configuration view, select the **Errands** pane to configure post-install errands or review the default settings. Post-install errands are scripts that automatically run after a product installs, before Ops Manager makes the product available for use. For more information about post-install errands, see the Understanding Lifecycle Errands topic.

   Note: By default, Ops Manager reruns errands even if they are not necessary due to settings left from a previous install. Leaving errands checked at all times can cause updates and other processes to take longer. To prevent an errand from running, deselect the checkbox for the errand in the **Settings** tab on the product tile before installing the product.



   The **Broker Registrar** checkbox is an example of an errand available for a product. When you select this checkbox, this errand registers service brokers with the Cloud Controller and also updates any broker URL and credential values that have changed since the previous registration.

8. In the Pending Changes view, click **Apply Changes** to start installation and run post-install lifecycle errands for the product.

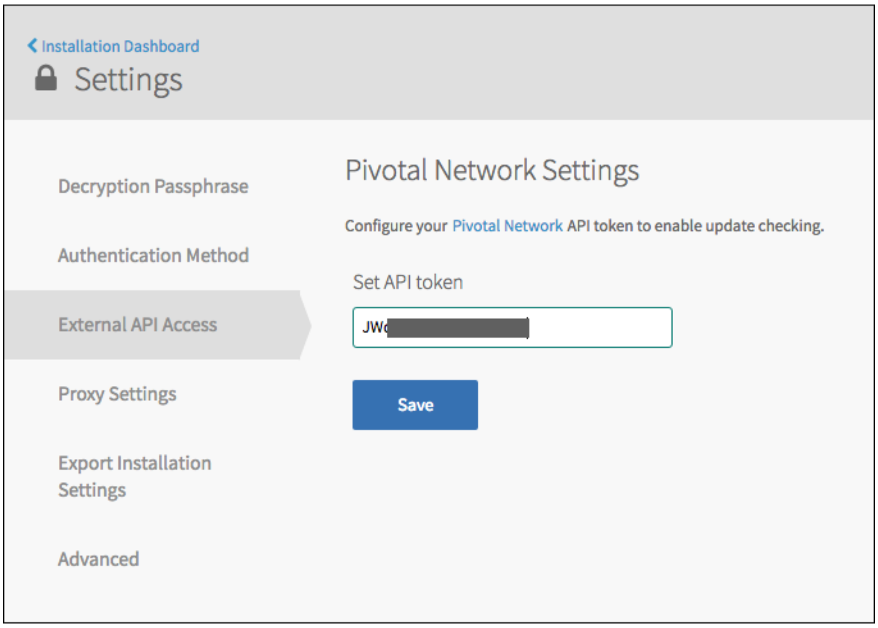
Using Pivotal Network API to Upgrade Products

Ops Manager provides a way to upgrade products by connecting your **Installation Dashboard** with Pivotal Network using a API token. Once you have uploaded a product, all subsequent product upgrades appear automatically in your **Installation Dashboard**.

Note: Using the Pivotal Network API is only available if you have access to the Internet since communication between Ops Manager and the Pivotal Network is necessary to import your products. If you are on an isolated network, do not save your API token.

1. Navigate to Pivotal Network and log in.
2. Click your user name, located in the upper top right side of the page.
3. Select **Edit Profile**.

4. In the **Edit Profile** tab, copy your **API Token**.
5. Navigate to your Ops Manager **Installation Dashboard** and log in.
6. Click your user name, located in the upper top right side of the page.
7. Select **Settings**.
8. In the **External API Access** tab, paste your **API Token**.

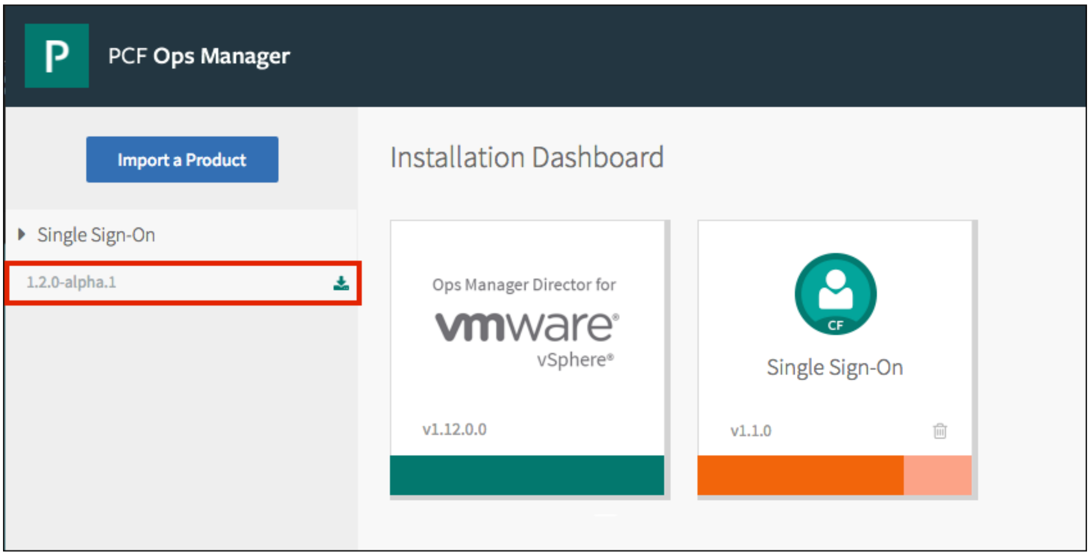


9. Click **Save**.

Update Existing Products

Once you save the Pivotal Network **API Token** to the Ops Manager **Installation Dashboard**, the latest versions of your existing products will appear in your **Installation Dashboard**. Upgrade your product to the latest version by following these instructions.

1. Locate and download the product version you want to upgrade to by clicking on the green download icon.



2. When the download is complete, refresh the page to use the product.

3. If necessary, configure the product.

4. In the Pending Changes view, click **Apply Changes**.

Applying Changes to Ops Manager Director

You can use the Ops Manager API to apply pending changes only to Ops Manager Director when you stage multiple products in a new installation or as part of an upgrade. For more information, see Applying Changes to Ops Manager Director.

Deleting a Product

1. From the Installation Dashboard, click the trash icon on a product tile to remove that product. In the **Delete Product** dialog box that appears, click **Confirm**.

   Note: You cannot delete the Ops Manager Director product.

2. In the Pending Changes view, click **Apply Changes**.
   After you delete a product, the product tile is removed from the installation and the Installation Dashboard. However, the product appears in the Available Products view.

Applying Changes to Ops Manager Director

Page last updated:

This topic describes how to apply pending changes only to the Ops Manager Director when you stage multiple products in a new installation or as part of an upgrade.

Overview

After you click **Apply Changes** on the Ops Manager Installation Dashboard, Ops Manager deploys the Ops Manager Director and all other products that have pending changes. You can optionally apply changes only to the Ops Manager Director using the Ops Manager API.

To deploy only the Ops Manager Director, you need to submit the `POST /api/v0/installations` API request. You must include the `deploy_products` parameter in this request and set the value of the parameter to `"none"`.

> **Note**: Submitting the `POST /api/v0/installations` API request is equivalent to clicking **Apply Changes** on the Ops Manager Installation Dashboard.

If you do not include `"deploy_products": "none"` in the `POST /api/v0/installations` request, Ops Manager deploys the Ops Manager Director and all other products with pending changes.

Apply Pending Changes to Ops Manager Director

To apply pending changes only to the Ops Manager Director, perform the steps below:

1. Ensure your Ops Manager Director tile is configured. The tile must be green in the Ops Manager Installation Dashboard.
2. Retrieve your authorization token to access the Ops Manager API. Refer to Using the Ops Manager API for the authentication instructions.
3. Submit the `POST /api/v0/installations` request with the `deploy_products` parameter set to `"none"`. See the following example:

```
$ curl "https://example.com/api/v0/installations" \
   -X POST \
   -H "Authorization: Bearer UAA-ACCESS-TOKEN" \
   -H "Content-Type: application/json" \
   -d '{
  "deploy_products": "none",
  "ignore_warnings": true
  }'
```

For more information about using the Ops Manager API, browse to the Ops Manager API documentation at `https://YOUR-OPS-MANAGER-FQDN/docs`.

Pivotal

Retrieving Credentials from Your Deployment

This topic describes how the credentials for your Pivotal Cloud Foundry (PCF) deployment are stored and how you can access them.

- **What credentials does PCF store?**
  - Many PCF components use credentials to authenticate connections, and PCF installations often have hundreds of active credentials. This includes certificates, VM credentials, and credentials for jobs running on the VMs.
- **Where does PCF store these credentials?**
  - PCF stores credentials in either the Ops Manager database or BOSH CredHub. In PCF v1.11 and later, the Ops Manager Director VM includes a co-located CredHub instance. Ops Manager, PAS, and service tiles running on PCF can use this CredHub instance to store their credentials. For example, in PCF v1.12, PAS began migrating its credentials to CredHub. See the PAS Release Notes for a full list.
- **When do I need to access these credentials?**:
  - You may need to access credentials for Ops Manager, PAS, and service tiles as part of regular administrative tasks in PCF, including troubleshooting. Many procedures in this documentation require you to retrieve credentials.
- **How can I retrieve credentials?**
  - The workflow for retrieving credentials depends on where they are stored. See the procedures below.

Retrieve Credentials Stored in BOSH CredHub

To retrieve credentials from CredHub using the Ops Manager API, do the following:

> **Note**: You can also retrieve credentials using the CredHub CLI from Ops Manager Director VM. For more information, see the CredHub CLI Readme.

1. Perform the procedures in the Using the Ops Manager API topic to authenticate and access the Ops Manager API.
2. Use the Ops Manager API to retrieve a list of deployed products:

   ```
   $ curl "https://OPS-MAN-FQDN/api/v0/deployed/products" \
   -X GET \
   -H "Authorization: Bearer UAA-ACCESS-TOKEN"
   ```

   Replace `UAA-ACCESS-TOKEN` with the access token recorded in the previous step.
3. In the response to the above request, locate the `guid` for the product from which you want to retrieve credentials. For example, if you want to retrieve PAS credentials, find the `installation_name` starting with `cf-` and copy its `guid`.
4. Run the following `curl` command to list the names of the credentials stored in CredHub for the product you selected. If you already know the name of the credential, you can skip this step.

   ```
   $ curl "https://OPS-MAN-FQDN/api/v0/deployed/products/PRODUCT-GUID/variables" \
   -X GET \
   -H "Authorization: Bearer UAA-ACCESS-TOKEN"
   ```

   Replace `PRODUCT-GUID` with the value of `guid` from the previous step.
5. Run the following `curl` command to view the credential:

   ```
   $ curl "https://OPS-MAN-FQDN/api/v0/deployed/products/PRODUCT-GUID/variables?name=VARIABLE-NAME" \
   -X GET \
   -H "Authorization: Bearer UAA-ACCESS-TOKEN"
   ```

   Replace `VARIABLE-NAME` with the name of the credential you want to retrieve.

Retrieve Credentials Stored in the Ops Manager Database

To retrieve credentials stored in the Ops Manager database and not CredHub, use the Ops Manager UI or API as outlined in the procedures below.

Retrieve Credentials Using the Ops Manager UI

1. From Ops Manager, select the product tile for which you want to retrieve credentials.

2. Click the **Credentials** tab.

3. Locate the credential that you need and click **Link to Credential**.

Retrieve Credentials Using the Ops Manager API

1. Perform the procedures in the Using the Ops Manager API topic to authenticate and access the Ops Manager API.

2. Use the Ops Manager API to retrieve a list of deployed products:

```
$ curl "https://OPS-MAN-FQDN/api/v0/deployed/products" \
-X GET \
-H "Authorization: Bearer UAA-ACCESS-TOKEN"
```

   Replace `UAA-ACCESS-TOKEN` with the access token recorded in the previous step.

3. In the response to the above request, locate the `guid` for the product from which you want to retrieve credentials. For example, if you want to retrieve PAS credentials, find the `installation_name` starting with `cf-` and copy its `guid`.

4. Run the following `curl` command to list references for the credentials stored in Ops Manager for the product you selected. If you already know the reference for the credential, you can skip this step.

```
$ curl "https://OPS-MAN-FQDN/api/v0/deployed/products/PRODUCT-GUID/credentials" \
-X GET \
-H "Authorization: Bearer UAA-ACCESS-TOKEN"
```

   Replace `PRODUCT-GUID` with the value of `guid` from the previous step.

5. Run the following `curl` command to view the credential:

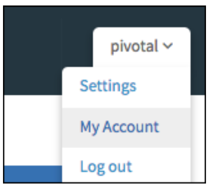
```
$ curl "https://OPS-MAN-FQDN/api/v0/deployed/products/PRODUCT-GUID/credentials/CREDENTIAL-REFERENCE" \
-X GET \
-H "Authorization: Bearer UAA-ACCESS-TOKEN"
```

   Replace `CREDENTIAL-REFERENCE` with the name of the credential you want to retrieve.

Changing Ops Manager Credentials

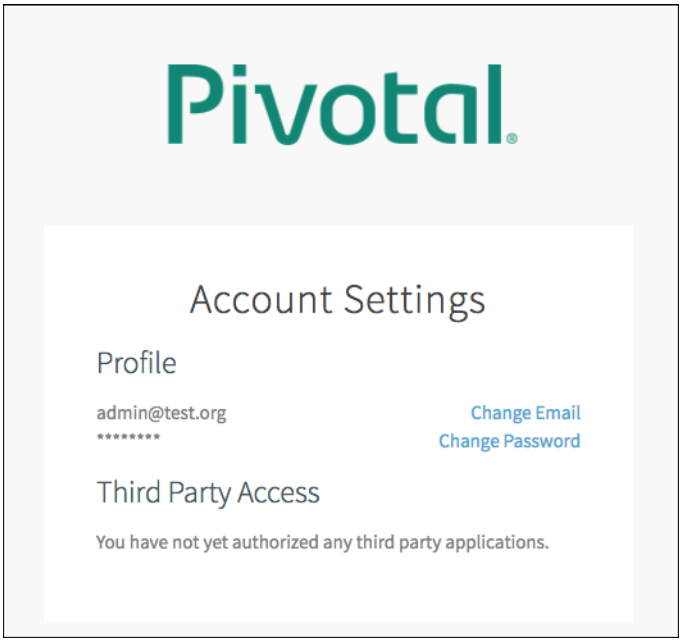
Ops Manager Password

1. Log in to Ops Manager and navigate to `My Account`. You can access this at `https://OPS-MAN-FQDN/uaa/profile`.



2. Navigate to `Change Password`. You can access this at `https://OPS-MAN-FQDN/uaa/change_password`.

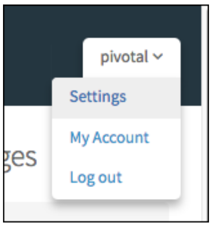
3. Enter your current password and a new password.



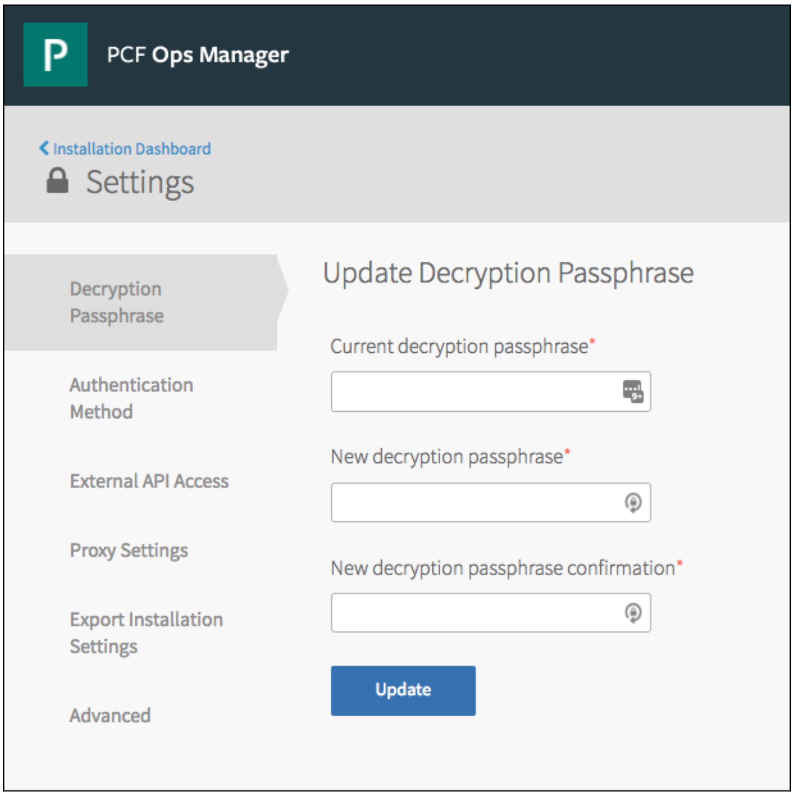
Ops Manager Decryption Passphrase

You must have the existing passphrase to update the decryption passphrase.

1. Log in to Ops Manager, and navigate to `Settings`. You can access this at `https://OPS-MAN-FQDN/encryption_passphrase/edit`.



2. In the `Decryption Passphrase` panel, enter your current decryption passphrase and the new decryption passphrase, then click **Save**.


PCF Ops Manager
Installation Dashboard
Settings
Decryption Passphrase
Authentication Method
External API Access
Proxy Settings
Export Installation Settings
Advanced
Update Decryption Passphrase
Current decryption passphrase*
New decryption passphrase*
New decryption passphrase confirmation*
Update

Understanding Floating Stemcells

Page last updated:

This topic describes how floating stemcells work in Pivotal Cloud Foundry (PCF) version 1.7 and later, and the consequences for upgrading product tiles in Ops Manager.

To increase the security of your deployment, all product tiles use floating stemcells by default. This enables tiles to automatically use the latest patched version of a stemcell.

When an operator upgrades a product tile, Ops Manager checks to see whether there is a new version of the stemcell. If an updated stemcell is available, Ops Manager installs the upgraded tile and all compatible product tiles in the deployment on the new stemcell. This ensures that when a vulnerability is discovered, PCF can quickly propagate a patched stemcell to all VMs in the deployment.

Operators can now perform certain deployment-wide updates, such as CVEs, by uploading a new stemcell instead of uploading `.pivotal` files for each tile, which reduces the time spent waiting for files to upload. Operators can upload new stemcells using the Ops Manager API or through a product tile in the Ops Manager Installation Dashboard.

However, operators who want to upgrade a single product tile may face significantly longer wait times, depending on the number of tiles in the deployment and the availability of a new stemcell.

Creating UAA Clients for BOSH Director

Page last updated:

This topic assumes you are using BOSH CLI v2.

This topic describes the process of creating a UAA client for the BOSH Director. You must create an automation client to run BOSH from a script or set up a continuous integration pipeline.

Local Authentication

To perform this procedure, the UAAC client must be installed on the Ops Manager virtual machine (VM).

1. Open a terminal and SSH into the Ops Manager VM by following the instructions for your IaaS in the SSH into Ops Manager topic.

2. Navigate to the Ops Manager **Installation Dashboard** and select the **Ops Manager Director** tile. In Ops Manager Director, click the **Status** tab, and copy the **Ops Manager Director** IP address.

3. Using the `uaac target` command, target Ops Manager Director UAA on port `8443` using the IP address you copied, and specify the location of the root certificate. The default location is `/var/tempest/workspaces/default/root_ca_certificate`.

```
$ uaac target https://OPS-DIRECTOR-IP:8443 --ca-cert \
/var/tempest/workspaces/default/root_ca_certificate

Target: https://10.85.16.4:8443
```

> **Note:** You can also curl or point your browser to the following endpoint to obtain the root certificate: `https://OPS-MANAGER-FQDN/api/v0/security/root_ca_certificate`

4. Log in to the Ops Manager Director UAA and retrieve the owner token. Perform the following step to obtain the values for `UAA-LOGIN-CLIENT-PASSWORD` and `UAA-ADMIN-CLIENT-PASSWORD`:
   - Select the **Ops Manager Director** tile from the Ops Manager **Installation Dashboard**.
   - Click the **Credentials** tab, and locate the entries for **Uaa Login Client Credentials** and **Uaa Admin User Credentials**.
   - For each entry, click **Link to Credential** to obtain the password.

```
$ uaac token owner get login -s UAA-LOGIN-CLIENT-PASSWORD
User name:  admin
Password:  UAA-ADMIN-CLIENT-PASSWORD

Successfully fetched token via owner password grant.
Target: https://10.85.16.4:8443
Context: admin, from client login
```

> **Note:** To obtain the password for the UAA login and admin clients, you can also curl or point your browser to the following endpoints: `https://OPS-MANAGER-FQDN/api/v0/deployed/director/credentials/uaa_login_client_credentials` and `https://OPS-MANAGER-FQDN/api/v0/deployed/director/credentials/uaa_admin_user_credentials`

5. Create a new UAA Client with `bosh.admin` privileges.

```
$ uaac client add ci --authorized_grant_types client_credentials \
--authorities bosh.admin --secret CI-SECRET

scope: uaa.none
client_id: ci
resource_ids: none
authorized_grant_types: client_credentials
autoapprove:
action: none
authorities: bosh.admin
name: ci
lastmodified: 1469727130702
id: ci
```

6. Set the client and secret as environment variables on the VM.

```
$ ubuntu@ip-10-0-0-12:~$ export BOSH_CLIENT=ci
$ ubuntu@ip-10-0-0-12:~$ export BOSH_CLIENT_SECRET=CI-SECRET
```

7. Set an alias for the BOSH Director environment. Replace `DIRECTOR-IP` with the IP address of your Ops Manager Director VM.

```
$ bosh alias-env MY-ENVIRONMENT-NAME -e DIRECTOR-IP \
--ca-cert /var/tempest/workspaces/default/root_ca_certificate
```

   You can now use the UAA client you created to run BOSH in automated or scripted environments, such as continuous integration pipelines.

SAML Authentication to the BOSH Director

Typically, there is no browser access to a BOSH Director in order to authenticate using SAML. Ops Manager provides an option to create UAA clients during SAML configuration so that BOSH can be automated via scripts and tooling.

1. Select **Provision an admin client in the Bosh UAA** when configuring Ops Manager for SAML.
2. After deploying Ops Manager Director (BOSH), click the Credentials tab in the Ops Manager Director tile.
3. Click the link for the **Uaa Bosh Client Credentials** to get the client name and secret.
4. Open a terminal and SSH into the Ops Manager VM. Follow the instructions for your SSH in the SSH into Ops Manager topic.
5. Set the client and secret as environment variables on the Ops Manager VM.

```
$ ubuntu@ip-10-0-0-12:~$ export BOSH_CLIENT=bosh_admin_client
$ ubuntu@ip-10-0-0-12:~$ export BOSH_CLIENT_SECRET=CLIENT_SECRET
```

6. Set an alias for the BOSH Director environment. Replace `DIRECTOR-IP` with the IP address of your Ops Manager Director VM.

```
$ bosh alias-env MY-ENVIRONMENT-NAME -e DIRECTOR-IP \
--ca-cert /var/tempest/workspaces/default/root_ca_certificate
```

Using Your Own Load Balancer

This guide describes how to use your own load balancer and forward traffic to your Pivotal Application Service (PAS) router IP address.

Pivotal Cloud Foundry (PCF) deploys with a single instance of HAProxy for use in lab and test environments. Production environments should use a highly-available customer-provided load balancing solution that does the following:

- Provides load balancing to each of the PCF Router IP addresses
- Supports SSL termination with wildcard DNS location
- Adds appropriate `x-forwarded-for` and `x-forwarded-proto` HTTP headers to incoming requests
- (**Optional**) Supports WebSockets

Note: Application logging with Loggregator requires WebSockets. To use another logging service, see the Using Third-Party Log Management Services topic.

Prerequisites

To integrate your own load balancer with PCF, you must ensure the following:

- WebSocket connections are not blocked for Loggregator functionality.
- The load balancer must be able to reach the Gorouter IP addresses.

Follow the instructions below to use your own load balancer.

Step 1: Deploy PCF Installation VM

Deploy a PCF Installation virtual machine. See the topic Deploying Operations Manager to vSphere for more information.

Step 2: Register PCF IP Address

In your load balancer, register the IP addresses that you assigned to PCF.

Step 3: Configure Pivotal Ops Manager and Ops Manager Director

Configure your Pivotal Operations Manager and Ops Manager Director as described in Installing Pivotal Cloud Foundry, then add PAS.

Do not click **Install** after adding PAS.

Step 4: Configure Networking

Complete the **Networking** configuration page in PAS. Load balancer configuration in PAS varies depending on which IaaS you are using for PCF. See the configuration procedure for your deployment IaaS:

- AWS
- Azure
- GCP
- OpenStack
- vSphere

Step 5: Finalize Changes

1. Return to the **Ops Manager Installation Dashboard**

2. Click **Install**.

Understanding Pivotal Cloud Foundry User Types

Page last updated:

This topic describes the types of users in a Pivotal Cloud Foundry (PCF) deployment, their roles and permissions, and who creates and manages their user accounts.

The users who run a PCF deployment and have admin privileges are operators. With Pivotal Application Service (PAS) installed to host apps, you add two more user types: PAS users who develop the apps and manage the development environment, and end users who just run the apps.

PCF distinguishes between these three user types and multiple user roles that exist within a single user type. Roles are assigned categories that more specifically define functions that a user can perform. A user may serve in more than one role at the same time.

Operators

Operators have the highest, admin-level permissions. We also refer to operators as Ops Manager admins and PAS admins because they perform an admin role within these contexts.

Tools and Tasks

Operators fulfill system administrator roles covering the entire PCF deployment. They work primarily with their IaaS and Ops Manager, to configure and maintain PAS component VMs. The component VMs, in turn, support the VMs that host applications. Typical operator tasks include:

- Deploying and configuring Ops Manager, PAS, and other product and service tiles.
- Maintaining and upgrading PCF deployments.
- Creating user accounts for PAS users and the orgs that PAS users work within.
- Creating service plans that define the access granted to end users.

User Accounts

When Ops Manager starts up for the first time, the operator specifies one of the following authentication systems for operator user accounts:

- Internal authentication, using a new UAA database that Ops Manager creates.
- External authentication, through an existing identity provider accessed through SAML protocol.

The operator can then use the UAAC to create more operator accounts.

PAS Users

PAS users are app developers, managers, and auditors who work within orgs and spaces, the virtual compartments within a deployment where PAS users can run apps and locally manage their roles and permissions.

A Role-Based Access Control (RBAC) system defines and maintains the different PAS user roles:

- Org Manager, Org Auditor, Org Billing Manager
- Space Manager, Space Developer, Space Auditor

The Orgs, Roles, Spaces, Permissions topic describes the PAS user roles, and what actions they can take within the orgs and spaces they belong to. Some of these permissions depend on the values of feature flags.

Tools

Space Developer users work with their software development tools and the apps deployed on host VMs.

All PAS users use system tools such as the Cloud Foundry Command Line Interface (cf CLI), PCF Metrics, and Apps Manager, a dashboard for managing PAS users, orgs, spaces, and apps.

User Accounts

When an operator configures PAS for the first time, they specify one of the following authentication systems for PAS user accounts:

1. Internal authentication, using a new UAA database created for PAS. This system-wide UAA differs from the Ops Manager internal UAA, which only stores Ops Manager Admin accounts.

2. External authentication, through an existing identity provider accessed through SAML or LDAP protocol.

In either case, PAS user role settings are saved internally in the Cloud Controller Database, separate from the internal or external user store.

Org and Space Managers then use Apps Manager to invite and manage additional PAS users within their orgs and spaces. PAS users with proper permissions can also use the cf CLI to assign user roles.

Operators can log into Apps Manager by using the **UAA Administrator User** credentials under the **Credentials** tab of the PAS tile. These UAA Admin credentials grant them the role of Org Manager within all orgs in the deployment. The UAA Admin can also use the UAAC to create new user accounts and the cf CLI to assign user roles.

End Users

End users are the people who log into and use the apps hosted on PAS. They do not interact directly with PAS components or interfaces. Any interactions or roles they perform within the apps are defined by the apps themselves, not Pivotal Application Service.

User Accounts and SSO

App developers can configure apps any way they want to grant end user access individually. In a deployment with Single Sign-On Service for Pivotal Cloud Foundry installed, they can also offer end users a single login that accesses multiple apps.

The Single Sign-On (SSO) service can save user account information in an external database accessed through SAML or LDAP, or in the internal PAS user store, along with PAS User accounts.

To make the SSO service available to developers, an operator creates service plans that give login access to specific groups of end users. A Space Manager then creates a local instance of the service plan, and registers apps with it. Apps registered to the plan instance then become available through SSO to all end users covered by the plan.

User Types Summary

The following table summarizes PCF user types, their roles, the tools they use, the System of Record (SOR) that stores their accounts, and what accounts they can provision.

User Type	Available Roles	Tools They Use	Account SOR	Accounts They Can Provision
Operators	Admin (UAA Admin, SSO Plan Admin, other system admins)	• IaaS UI • PivNet • Ops Manager • cf CLI • UAA CLI (UAAC) • SSO Dashboard • Marketplace	Ops Manager user store through UAA *or* External store through SAML	Operators and PAS Users
PAS Users	• UAA Administrator • Org Manager • Org Auditor • Org Billing Manager • Space Manager • Space Developer • Space Auditor	• cf CLI • CAPI • Apps Manager • PCF Metrics • Marketplace	PAS user store through UAA *or* External store through SAML or LDAP	PAS Users within permitted orgs and spaces, and End Users

End Users	Defined by apps they use	Hosted apps	Individual apps *or* PAS user store through SSO	

Creating and Managing Ops Manager User Accounts

Page last updated:

Pivotal Cloud Foundry supports multiple user accounts in Ops Manager. A User Account and Authentication (UAA) module co-located on the Ops Manager VM manages access permissions to Ops Manager.

When Ops Manager boots for the first time, you create an admin user. However, you do not create additional users through the Ops Manager web interface. If you want to create additional users who can log into Ops Manager, you must use the UAA API, either through `curl` or the UAA Command Line Client (UAAC).

> **Note:** You can only manage users on the Ops Manager UAA module if you chose to use Internal Authentication instead of an external Identity Provider when configuring Ops Manager.

Follow these steps to add or remove users via the UAAC. If you do not already have the UAAC installed, run `gem install cf-uaac` from a terminal window.

Adding Users to Ops Manager

1. Target your Ops Manager UAA:

```
$ uaac target https://YOUR-OPSMAN-FQDN/uaa/
```

2. Get your token:

```
$ uaac token owner get
Client ID: opsman
Client Secret: [Press Enter]
Username: Admin
Password: *******

Successfully fetched token via client credentials grant.
Target https://YOUR-OPSMAN-FQDN/uaa/
```

3. Add a user:

```
$ uaac user add YOUR-USER-NAME -p YOUR-USER-PASSWORD --emails YOUR-USER-EMAIL@EXAMPLE.COM
```

Removing Users from Ops Manager

1. Target your Ops Manager UAA:

```
$ uaac target https://YOUR-OPSMAN-FQDN/uaa/
```

2. Get your token:

```
$ uaac token owner get
Client ID: opsman
Client Secret: [Press Enter]
Username: Admin
Password: *******

Successfully fetched token via client credentials grant.
Target https://YOUR-OPSMAN-FQDN/uaa/
```

3. Delete a user:

```
$ uaac user delete YOUR-USER-NAME
```

Configuring Role-Based Access Control (RBAC) in Ops Manager

Page last updated:

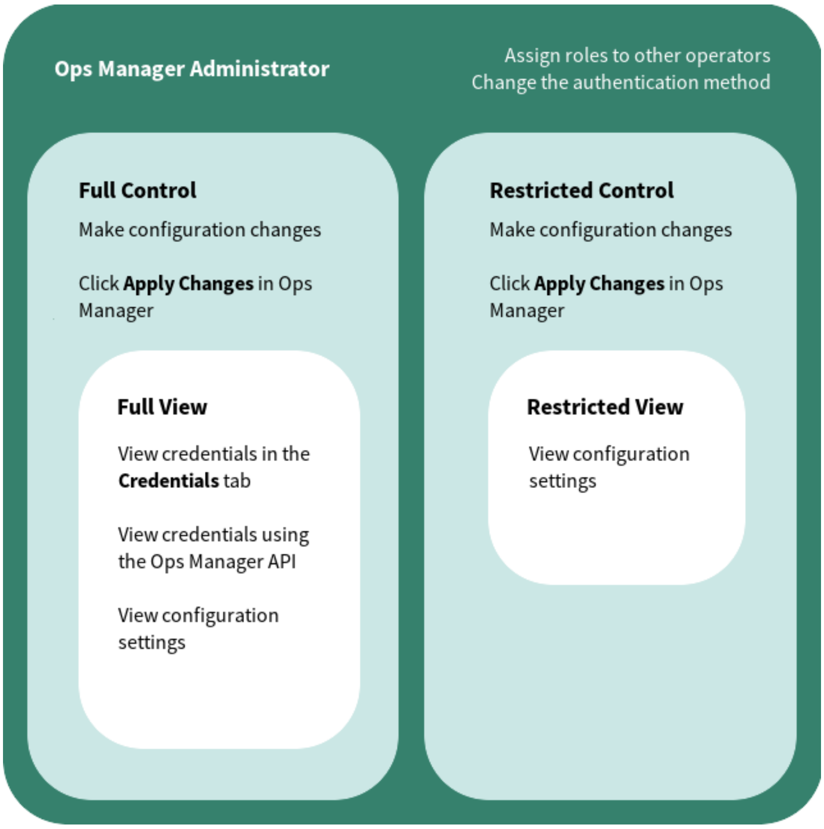
This topic describes how to customize role-based access control (RBAC) in Ops Manager. Use RBAC to manage which operators in your organization can make deployment changes, view credentials, and manage user roles in Ops Manager.

For information about configuring Ops Manager to use internal authentication or SAML authentication, refer to the Ops Manager configuration topic for your IaaS:

- Configuring Ops Manager on AWS
- Configuring Ops Manager on Azure
- Configuring Ops Manager on GCP
- Configuring Ops Manager on OpenStack
- Configuring Ops Manager on vSphere

Understand Roles in Ops Manager

You can assign the following roles to determine which operators in your organization make deployment changes, view credentials, and manage user roles in Ops Manager:



Ops Manager administrators can use the roles defined in the diagram above to meet the security needs of their organization. The roles provide a range of privileges that are appropriate for different types of users. For example, assign either **Restricted Control** or **Restricted View** to an operator to prevent access to all Ops Manager credentials.

See the following table for more information about each role:

Ops Manager Role	Role Definition	UAA Scope
Ops Manager Administrator	Administrators can make configuration changes in Ops Manager, view credentials in the **Credentials** tab and Ops Manager API endpoints, change the authentication method, and assign roles to other operators.	`opsman.admin`
	Operators can make configuration changes in Ops Manager, click **Apply Changes**, and view	

Ops Manager Role	Role Definition	UAA Scope
Full Control	credentials in the **Credentials** tab and Ops Manager API endpoints.	`opsman.full_control`
Restricted Control	Operators can make configuration changes in Ops Manager and click **Apply Changes**. They cannot view credentials in the **Credentials** tab or Ops Manager API endpoints.	`opsman.restricted_control`
Full View	Operators can view Ops Manager configuration settings and view credentials in the **Credentials** tab and Ops Manager API endpoints. They cannot make configuration changes in Ops Manager.	`opsman.full_view`
Restricted View	Operators can view Ops Manager configuration settings. They cannot make configuration changes or view credentials in the **Credentials** tab or Ops Manager API endpoints.	`opsman.restricted_view`

When you install a new Ops Manager instance, all existing users have the Ops Manager Administrator role by default.

To assign one of the above roles to an operator, follow the procedure for granting access using either internal authentication or SAML authentication.

Enable RBAC in Ops Manager

When you install a new instance of Ops Manager, RBAC is enabled by default.

If you upgrade from an older Ops Manager instance, you must enable RBAC and assign roles to users before they can access Ops Manager. If you do not assign any roles to a user, they cannot log in to Ops Manager.

WARNING: Do not assign roles before you enable RBAC.

Enable RBAC with Internal Authentication

If you are upgrading from an older version of Ops Manager and use internal authentication, perform the following steps to enable RBAC:

1. Log in to Ops Manager and click the Ops Manager tile.
2. In the **Settings** tab, click **Advanced**.
3. Click **Enable RBAC**. When the confirmation dialog box appears, click **Confirm and Logout**.

   Note: Enabling RBAC is permanent. You cannot undo this action. When you upgrade Ops Manager, your RBAC settings remain configured.

Enable RBAC with SAML Authentication

If you are upgrading from an older version of Ops Manager and use SAML authentication, perform the steps in this section to enable RBAC. To enable RBAC in Ops Manager when using SAML authentication, you must configure groups in SAML for admins and non-admins and then map the admin group to Ops Manager.

Step 1: Configure SAML Groups

Perform the following steps to gather information from your SAML dashboard:

1. Log in to your SAML provider dashboard.
2. Create or identify the name of the SAML group that contains Ops Manager admin users.
3. Identify the groups attribute tag you configured for your SAML server.

Step 2: Enable RBAC in Ops Manager

Perform the following steps to configure Ops Manager to recognize your SAML admin user group:

Note: When RBAC is enabled, only users with the Ops Manager Administrator role can edit SAML configuration.

1. Log in to Ops Manager.
2. In the **Settings** tab, click **RBAC Configuration**.
3. Enter the name of the SAML group that contains Ops Manager admin users in the **SAML Admin Group** field.
4. Enter the groups attribute tag for your SAML server in the **Groups Attribute** field.
5. In the **Settings** tab, click **Advanced**.
6. Click **Enable RBAC**. When the confirmation dialog box appears, click **Confirm and Logout**.

> **Note**: Enabling RBAC is permanent. You cannot undo this action. When you upgrade Ops Manager, your RBAC settings remain configured.

Manage RBAC Roles in Ops Manager

You can assign the roles defined in Understanding Roles in Ops Manager to determine which operators in your organization make deployment changes, view credentials, and manage user roles in Ops Manager.

Manage Roles with Internal Authentication

If you configured Ops Manager to use internal authentication, perform the steps in this section to configure roles using the UAA Command Line Interface (UAAC).

1. Target your UAA server and log in as an admin:

```
uaac target https://YOUR-OPSMAN-DOMAIN/uaa
uaac token owner get
```

2. When prompted, enter the following credentials. Enter `opsman` for **Client ID** and leave **Client secret** blank, then enter your username and password:

```
Client ID: opsman
Client secret:
User name: USERNAME
Password: YOUR-PASSWORD
```

3. **(Optional)** If you are installing a new Ops Manager instance, create users by following the procedure in the Creating and Managing Users with the UAA CLI (UAAC) topic.
4. Assign one of the following roles to a user, replacing `USERNAME` with their username.
   - **Ops Manager Administrator**:

   ```
   uaac member add opsman.admin USERNAME
   ```

   - **Full Control**:

   ```
   uaac member add opsman.full_control USERNAME
   ```

   - **Restricted Control**:

   ```
   uaac member add opsman.restricted_control USERNAME
   ```

   - **Full View**:

   ```
   uaac member add opsman.full_view USERNAME
   ```

   - **Restricted View**:

   ```
   uaac member add opsman.restricted_view USERNAME
   ```

Manage Roles with SAML Authentication

If you configured Ops Manager with SAML authentication, perform the steps in this section to assign non-admin user roles using UAAC.

1. Target your UAA server and log in as an admin:

   ```
   uaac target https://YOUR-OPSMAN-DOMAIN/uaa
   uaac token sso get
   ```

2. When prompted, enter **Client ID** and **Passcode**, leaving **Client secret** blank:

   ```
   Client ID: opsman
   Client secret:
   Passcode (from http://YOUR-OPSMAN-DOMAIN/uaa/passcode): YOUR-UAA-PASSCODE
   ```

3. Run the following command:

   ```
   uaac group map SAML-GROUP --name 'OPSMAN-SCOPE' --origin 'external-saml-provider'
   ```

   Replace the placeholder text as follows:
   - `SAML-GROUP`: Replace with name of the SAML group the user belongs to.
   - `OPSMAN-SCOPE`: Replace with an Ops Manager UAA scope. Refer to the table in Understand Roles in Ops Manager to determine which UAA scope to use.

4. Add new and existing users to the appropriate SAML groups in the SAML provider dashboard. Users must log out of both Ops Manager and the SAML provider for role changes to take effect.

Pivotal

Creating New PAS User Accounts

Page last updated:

When you first deploy your Pivotal Application Service (PAS), there is only one user: an administrator. At this point, you can add accounts for new users who can then push applications using the Cloud Foundry Command Line Interface (cf CLI).

How to add users depends on whether or not you have SMTP enabled, as described in the options below.

Option 1: Adding New Users when SMTP is Enabled

If you have enabled SMTP, your users can sign up for accounts and create their own orgs. They do this using the Pivotal Cloud Foundry (PCF) Apps Manager, a self-service tool for managing organizations, users, applications, and application spaces.

Instruct users to complete the following steps to log in and get started using the Apps Manager.

1. Browse to `apps.YOUR-SYSTEM-DOMAIN`. Refer to PAS **Domains** to locate your system domain.

2. Select **Create an Account**.

3. Enter your email address and click **Create an Account**. You will receive an email from the Apps Manager when your account is ready.

4. When you receive the new account email, follow the link in the email to complete your registration.

5. You will be asked to choose your organization name.

You now have access to the Apps Manager. Refer to the Apps Manager documentation at docs.pivotal.io for more information about using the Apps Manager.

Option 2: Adding New Users when SMTP is Not Enabled

If you have not enabled SMTP, only an administrator can create new users, and there is no self-service facility for users to sign up for accounts or create orgs.

The administrator creates users with the cf CLI. See Creating and Managing Users with the cf CLI.

Return to the Installing Pivotal Cloud Foundry Guide

Logging in to Apps Manager

Page last updated:

Log in as Admin User

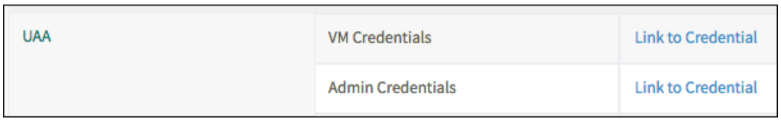
Complete the following steps to log in to Apps Manager as the Admin user:

1. If you do not know the system domain for the deployment, then select Pivotal Application Service (PAS) **Settings > Domains** to locate the configured system domain.

2. Open a browser and navigate to `apps.YOUR-SYSTEM-DOMAIN`. For example, if the system domain is `system.example.com`, then point your browser to `apps.system.example.com`.

3. If you have enabled Pivotal Account, the browser redirects to `login.YOUR-SYSTEM-DOMAIN`. For example, `login.system.example.com`.

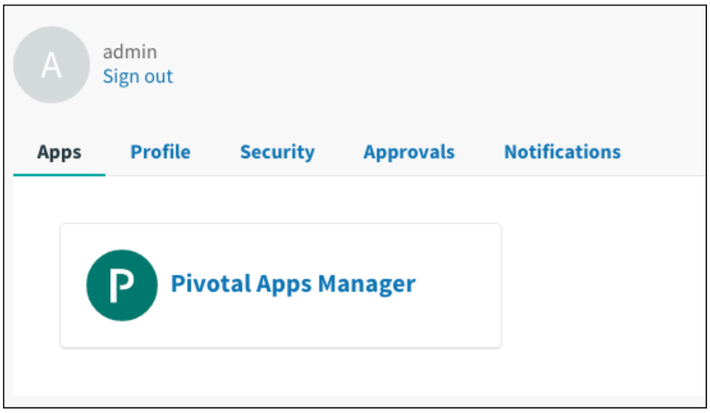
4. Log in using UAA credentials for the Admin user. To obtain these credentials, refer to PAS **Credentials > UAA > Admin Credentials**.



5. After you log in, Apps Manager appears.

Access through Pivotal Account Page

You can also access Apps Manager through the Pivotal Account interface by logging in to `login.YOUR-SYSTEM-DOMAIN`. Then, select the **Apps** tab, and click on **Pivotal Apps Manager** to open Apps Manager.



See Enabling Pivotal Account for more information about managing the account page.

Adding Existing SAML or LDAP Users to a PCF Deployment

This topic describes the procedure for adding existing SAML or LDAP users to a Pivotal Cloud Foundry (PCF) deployment enabled with SAML or LDAP.

The following two ways exist to add existing SAML or LDAP users to your PCF deployment:

- Option 1: Import Users in Bulk
- Option 2: Add Users Manually

Prerequisites

You must have the following to perform the procedures in this topic:

- Admin access to the Ops Manager Installation Dashboard for your PCF deployment
- The Cloud Foundry Command Line Interface (cf CLI) v6.23.0 or later

Option 1: Import Users in Bulk

You can import SAML or LDAP users in bulk by using the UAA Bulk Import Tool. See the UAA Users Import README for instructions about installing and using the tool.

Option 2: Add Users Manually

Perform the procedures below to add existing SAML or LDAP users to your PCF deployment manually.

Step 1: Create User

Perform the following steps to add a SAML or LDAP user:

1. Run `cf target https://api.YOUR-SYSTEM-DOMAIN` to target the API endpoint for your PCF deployment. Replace `YOUR-SYSTEM-DOMAIN` with your system domain. For example:

```
$ cf target https://api.example.com
```

2. Run `cf login` and provide credentials for an account with the Admin user role:

```
$ cf login
```

3. Run `cf create-user EXAMPLE-USERNAME --origin YOUR-PROVIDER-NAME` to create the user in UAA. Replace `EXAMPLE-USERNAME` with the username of the SAML or LDAP user you wish to add, and select one of the options below:
   - For LDAP, replace `YOUR-PROVIDER-NAME` with `ldap`. For example:

   ```
   $ cf create-user j.smith@example.com --origin ldap
   ```

   - For SAML, replace `YOUR-PROVIDER-NAME` with the name of the SAML provider you provided when configuring Ops Manager. For example:

   ```
   $ cf create-user j.smith@example.com --origin example-saml-provider
   ```

Step 2: Associate User with Org or Space Role

After creating the SAML or LDAP user, you must associate the user with either an Org or Space role.

For more information about roles, see the Roles and Permissions section of the *Orgs, Spaces, Roles, and Permissions* topic.

Associate User with Org Role

Run `cf set-org-role USERNAME YOUR-ORG ROLE` to associate the SAML or LDAP user with an Org role. Replace `USERNAME` with the name of the SAML or LDAP user, and replace `YOUR-ORG` with the name of your Org.

For `ROLE`, enter one of the following:

- `OrgManager`: Org Managers can invite and manage users, select and change plans, and set spending limits.
- `BillingManager`: Billing Managers can create and manage the billing account and payment information.
- `OrgAuditor`: Org Auditors have read-only access to Org information and reports.

Example:

```
$ cf set-org-role j.smith@example.com my-org OrgManager
```

Associate User with Space Role

Run `cf set-space-role USERNAME YOUR-ORG YOUR-SPACE ROLE` to associate the SAML or LDAP user with an Org role. Replace `USERNAME` with the name of the SAML or LDAP user, replace `YOUR-ORG` with the name of your Org, and `YOUR-SPACE` with the name of a Space in your Org.

For `ROLE`, enter one of the following:

- `SpaceManager`: Space Managers can invite and manage users, and enable features for a given Space.
- `SpaceDeveloper`: Space Developers can create and manage apps and services, and see logs and reports.
- `SpaceAuditor`: Space Auditors can view logs, reports, and settings on this Space.

Example:

```
$ cf set-space-role j.smith@example.com my-org my-space SpaceDeveloper
```

Modifying Your Ops Manager Installation and Product Template Files

Page last updated:

This topic describes how to modify your Ops Manager installation by decrypting and editing the YAML files that Ops Manager uses to store configuration data. Operators can use these procedures to view and change values that they cannot access through the Ops Manager web interface. They can also modify the product templates that Ops Manager uses to create forms and obtain user input.

Operators may want to modify the Ops Manager installation and product template files for a number of reasons, including the following:

- To change the User Account and Authentication (UAA) admin password of their deployment
- To retrieve key values
- To migrate content across different Pivotal Cloud Foundry (PCF) releases

WARNING: Be careful when making changes to your Ops Manager installation and product template files. Use spaces instead of tabs, and remember that YAML files use whitespace as a delimiter. Finally, Pivotal does not officially support these procedures, so use them at your own risk.

Understand Installation and Product Template Files

During the installation process, Ops Manager combines information from the installation and product template files to generate the manifests that define your deployment.

- **Installation file**: PCF stores user-entered data and automatically generated values for Ops Manager in an installation YAML file on the Ops Manager virtual machine (VM). PCF encrypts and stores this file in the directory `/var/tempest/workspaces/default`. You must decrypt this file to view the contents, edit them as necessary, then re-encrypt them.
- **Product templates**: Ops Manager uses product templates to create forms and obtain user input. The `job_types` and `property_blueprint` key-value pairs in a product template determine how the `jobs` and `properties` sections display in the installation file. Ops Manager stores product templates as YAML files in the directory `/var/tempest/workspaces/default/metadata` on the Ops Manager VM. These files are not encrypted, so you can edit them without decrypting. User input does not alter these files.

Note: Upgrading Ops Manager may eliminate your changes to the installation and product template files.

Modify the Installation File

Perform the following steps to locate, decrypt, and edit your Ops Manager installation file:

1. SSH into the Ops Manager VM by following the steps in the SSH into Ops Manager section of the *Advanced Troubleshooting with the BOSH CLI* topic.
2. `cd` into the scripts directory:

```
$ cd /home/tempest-web/tempest/web/scripts/
```

3. Run the following command to decrypt the installation YAML file and make a temporary copy of the decrypted file. When prompted for a passphrase, enter the decryption passphrase you created when you launched Ops Manager for the first time:

```
$ sudo -u tempest-web ./decrypt /var/tempest/workspaces/default/installation.yml /tmp/installation.yml
```

4. Open `/tmp/installation.yml` to view or edit values.
5. If you plan to make changes, make a backup of the original installation YAML file:

```
$ cp /var/tempest/workspaces/default/installation.yml ~/installation-orig.yml
```

6. If you have made changes to your copy of the installation YAML file, you must encrypt it and overwrite the original with it:

```
$ sudo -u tempest-web RAILS_ENV=production /home/tempest-web/tempest/web/scripts/encrypt /tmp/installation.yml /var/tempest/workspaces/default/installation.yml
```

When prompted, enter a passphrase.

7. Delete the temporary copy of the decrypted file:

```
$ rm /tmp/installation.yml
```

8. Restart the Ops Manager web interface:

```
$ sudo service tempest-web stop && sudo service tempest-web start
```

9. Navigate to Ops Manager in a browser and enter your decryption passphrase.
10. Log in to Ops Manager and click **Apply Changes**.
11. If Ops Manager cannot load your changes, see the Revert To Your Backup section of this topic to restore your previous settings.

Modify Product Template Files

Perform the following steps to locate and edit your Ops Manager product template files:

1. SSH into the Ops Manager VM by following the steps in the SSH into Ops Manager section of the *Advanced Troubleshooting with the BOSH CLI* topic.
2. On the Ops Manager VM, navigate to the `/var/tempest/workspaces/default/metadata` directory.

```
$ cd /var/tempest/workspaces/default/metadata
```

3. The `/var/tempest/workspaces/default/metadata` directory contains the product templates as YAML files. If you plan to make changes, make a backup of the original product template YAML file:

```
$ cp /var/tempest/workspace/default/metadata/YOUR-PRODUCT-TEMPLATE.yml ~/YOUR-PRODUCT-TEMPLATE-orig.yml
```

4. Open and edit the product template YAML file as necessary. For more information about product templates, see the Product Template Reference topic.
5. Navigate to Ops Manager to see your changes.
6. If Ops Manager cannot load your changes, see the Revert To Your Backup section of this to restore your previous settings.

Revert to Your Backup

Perform the following steps to revert to your backup of an installation or product template file:

1. SSH into the Ops Manager VM by following the steps in the SSH into Ops Manager section of the *Advanced Troubleshooting with the BOSH CLI* topic.
2. Overwrite the modified file with the backup:
   - For the installation file, run the following command:

   ```
   $ cp ~/installation-orig.yml /var/tempest/workspaces/default/installation.yml
   ```

   - For a product template file, run the following command:

   ```
   $ cp ~/YOUR-PRODUCT-TEMPLATE-orig.yml /var/tempest/workspaces/default/metadata/YOUR-PRODUCT-TEMPLATE.yml
   ```

3. Restart the Ops Manager web interface:

```
$ sudo service tempest-web stop && sudo service tempest-web start
```

4. Navigate to Ops Manager in a browser and enter your decryption passphrase.
5. Log in to Ops Manager and click **Apply Changes**.

Managing Errands in Ops Manager

Page last updated:

This topic describes product errands and how to configure them in Pivotal Cloud Foundry (PCF) Ops Manager.

Errands are scripts that can run at the beginning and at the end of an installed product's availability time. You can use Ops Manager to adjust whether and when these errands run.

Product tiles include two types of errands:

- **Post-deploy errands** run after the product installs but before Ops Manager makes the product available for use. One example is an errand that publishes a newly-installed service to the Services Marketplace.
- **Pre-delete errands** run after an operator chooses to delete the product but before Ops Manager actually deletes it. One example is a clean-up task that removes all data objects used by the errand.

When you click **Apply Changes** in Ops Manager, BOSH either creates a VM for each errand that runs or co-locates errands on existing VMs. Tile developers determine where BOSH deploys the errands for their product.

Pivotal Application Service (PAS) provides several post-deploy errands including smoke tests, Apps Manager, notification, Pivotal Account, and autoscaling errands. For more information about PAS errands, see the *Deploying PAS* topic for the platform where you are deploying PAS. For example, if you are deploying PAS on GCP, see Deploying PAS on GCP.

For information about the errands associated with any other PCF product, see the documentation provided with the product tile.

Errand Run Rules

Operators can configure three different run rules for errands: **On**, **Off**, and **When Changed**. These rules control when Ops Manager executes the errand.

When the errand is configured to be...	Then the errand...
On	always runs, even when there are no changes to the product manifest.
Off	never runs.
When Changed	runs only if the errand configuration in the associated product deployment manifest has changed since the last time the errand succeeded. If there are no changes to the errand configuration of the product manifest **and** the most recent run of the errand succeeded, then the errand does not run.

Ops Manager Defaults and Tile Defaults

By default, Ops Manager applies the **When Changed** rule to post-deploy errands and the **On** rule to pre-delete errands.

For any errand, the tile developer can override these Ops Manager defaults with their own tile-specific defaults defined in the tile property blueprints.

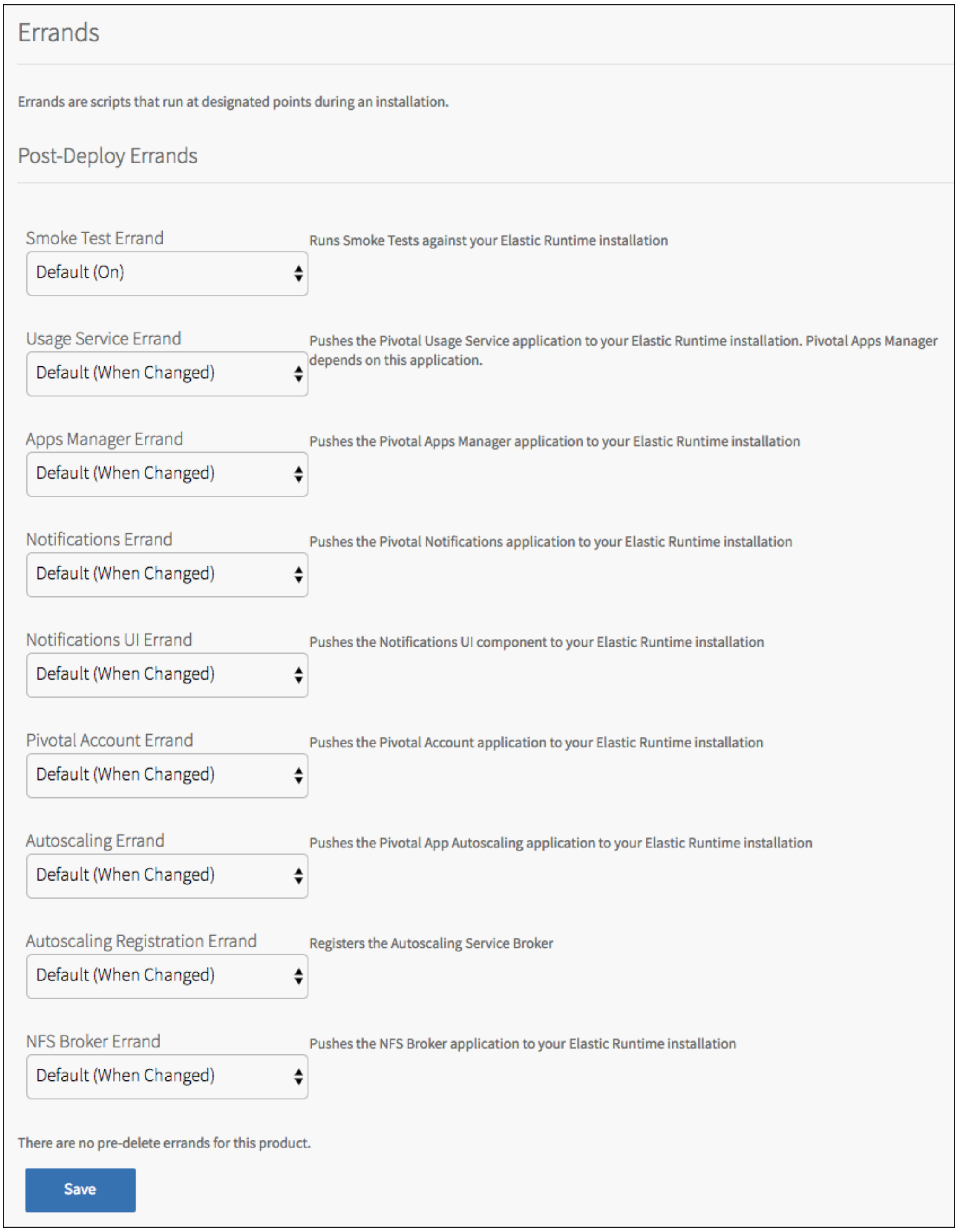
Configure Run Rules in Ops Manager

You can configure the run rules for errands in two places in Ops Manager. The Errands pane saves your configuration and applies the configuration to future installations. The Pending Changes view applies the rules only to the next time you run an Ops Manager install, without saving them.

Errands Pane: Persistent Rules

Product tiles for PAS and other PCF products have an **Errands** pane that configures the run rules for the product's errands and saves the settings for later.

The **Errands** pane lists all errands for the product and lets you select one of four run rule choices for each: **On**, **Off**, **When Changed**, and a default option **Default (On)**, **Default (Off)**, or **Default (When Changed)**. The default option differs depending on the errand, and reflects the default setting used by Ops Manager for the errand or any tile-specific default that overrides it.



Follow these steps to configure the run rules for a tile:

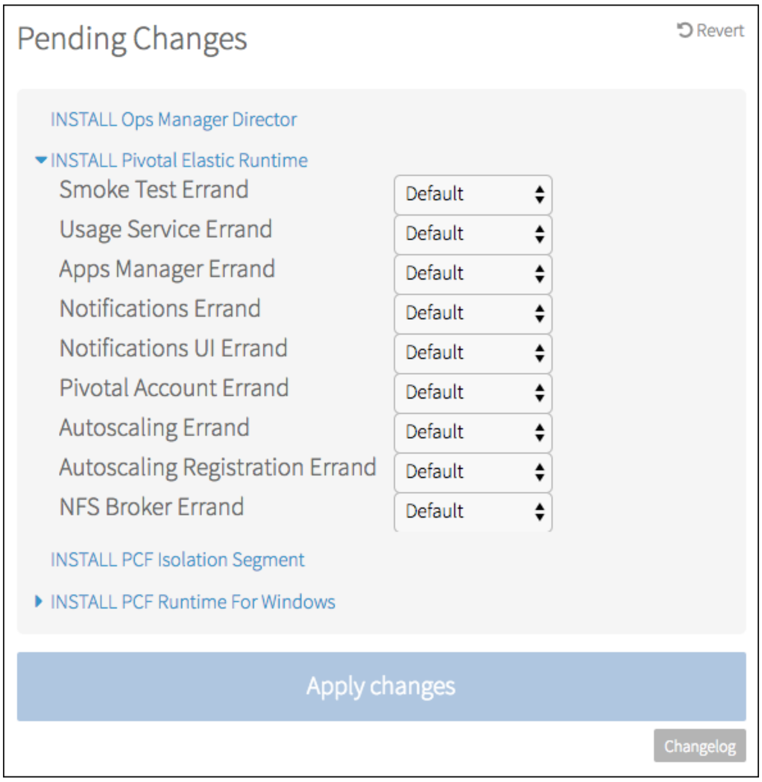
1. Navigate to Ops Manager and click the tile to open it.
2. Under the **Settings** tab, open the **Errands** pane.
3. Use the drop-down menus to configure the run rule choice for each errand: **On**, **Off**, **When Changed**, or the **Default** option.
4. Click **Save** to save the configuration values and return to the Installation Dashboard.
5. Click **Apply Changes** to redeploy the tile with the new settings.

Pending Changes: One-Time Rules

Ops Manager lets you quickly configure one-time errand run rules for any product queued up for installation:

1. Navigate to Ops Manager. The **Pending Changes** section at top right shows products that Ops Manager has yet to install or update.

2. Under **Pending Changes**, click the product you wish to configure. A list of errands associated with the product appears.



3. Use the drop-down menus to configure the run rule choice for each errand: **On**, **Off**, **When Changed**, or **Default**. Ops Manager applies these settings once you click **Apply changes** to install the product, but does not save the settings for future installations.

4. Click **Apply Changes** to redeploy the tile.

Related Links

If you are a product developer and want to learn more about adding errands to your product tile for PCF, see the Errands topic in the *PCF Tile Developer Guide*.

Monitoring PCF VMs from Ops Manager and vSphere

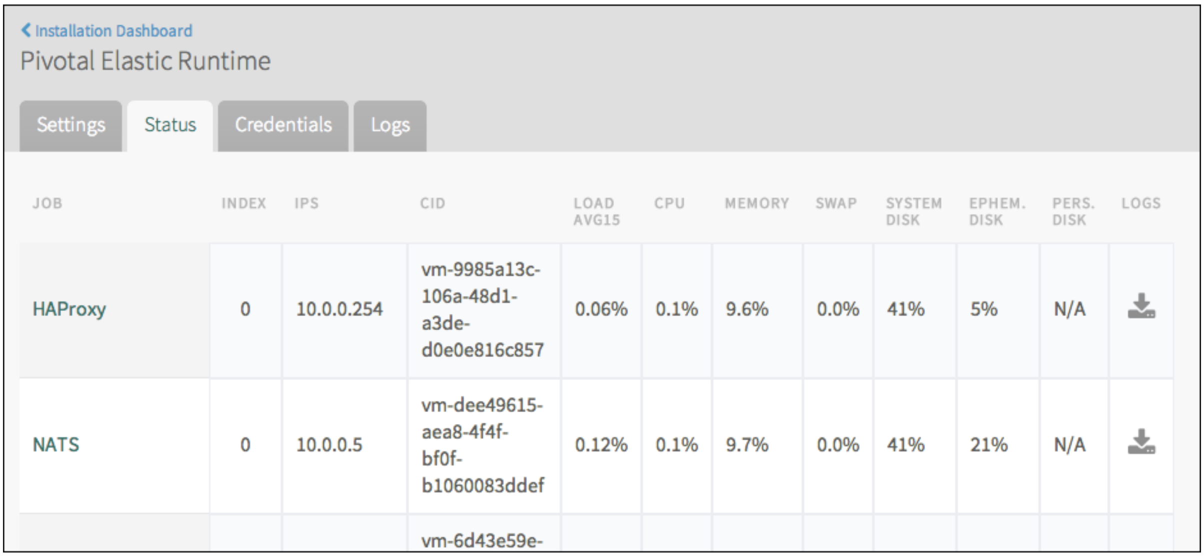
Page last updated:

This topic describes how to check current VM status in Pivotal Cloud Foundry (PCF) Ops Manager and how to use the vSphere client to perform ongoing monitoring and alerting of PCF VMs.

For a complete guide to monitoring PCF, see Monitoring Pivotal Cloud Foundry.

Monitoring VMs Using the Ops Manager Interface

Click any product tile and select the **Status** tab to view monitoring information.



The columns display the following information:

VM Data Point	Details
Job	Each job represents a component running on one or more VMs that Ops Manager deployed.
Index	For jobs that run across multiple VMs, the **index** value indicates the order in which the job VMs were deployed. For jobs that run on only one VM, the VM has an **index** value of `0`.
IPs	IP address of the job VM.
CID	Uniquely identifies the VM.
Load Avg15	CPU load average over 15 minutes.
CPU	Current CPU usage.
Memory	Current memory usage.
Swap	Swap file percentage.
System Disk	System disk space usage.
Ephem. Disk	Ephemeral disk space usage.
Pers. Disk	Persistent disk space usage.
Logs	Download link for the most recent log files.

Operations Manager VM Disk Space

The Ops Manager stores its logs on the Ops Manager VM in the `/tmp` directory.

> **Note**: The logs collect over time and do not self-delete. To prevent the VM from running out of disk space, restart the VM to clear the log entries from `/tmp`.

Monitoring in vSphere

To monitor VMs using the vSphere client:

1. Connect to a vCenter Server instance using the vSphere client.
2. Navigate to the **Hosts And Clusters** or **VMs And Templates** inventory view.
3. In the inventory tree, select a virtual machine.
4. Select the **Performance** tab from the content pane on the right.

VMware vSphere Server provides alarms that monitor VMs, as well as clusters, hosts, datacenters, datastores, networks, and licensing. To view preconfigured alarms, including disk usage alarms, related to a particular VM:

1. In the vSphere client, select the VM you want to monitor.
2. At the bottom left of the client window, click **Alarms**.
3. If a VM starts to run out of disk space, an alarm appears in the bottom panel.

Cloud Foundry Concepts

Cloud Foundry is an open platform as a service, providing a choice of clouds, developer frameworks, and application services. Cloud Foundry makes it faster and easier to build, test, deploy and scale applications. It is an open source project and is available through a variety of private cloud distributions and public cloud instances.

This guide presents an overview of how Cloud Foundry works and a discussion of key concepts. Refer to this guide to learn more about Cloud Foundry fundamentals.

General Concepts

- Cloud Foundry Overview
- How Applications are Staged
- High Availability in Cloud Foundry
- Orgs, Spaces, Roles, and Permissions
- Understanding Cloud Foundry Security
- Understanding Container Security
- Understanding Container-to-Container Networking
- Understanding Application Security Groups

Architecture

- Cloud Foundry Components
- Component: Cloud Controller
- Component: Messaging (NATS)
- Component: Gorouter
- Component: User Account and Authentication (UAA) Server
- Component: Garden
- Component: HTTP Routing
- Component: Droplet Execution Agent (for cf-release v261 and earlier)
- Component: DEA Placement Algorithm (for cf-release v261 and earlier)
- Component: Warden (for cf-release v261 and earlier)

Diego

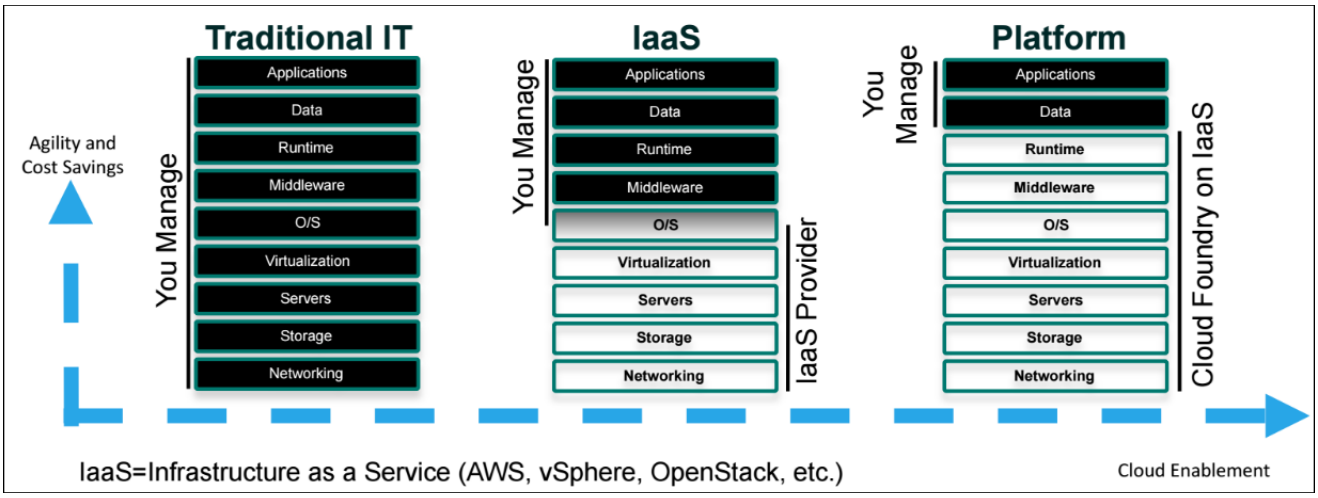
- Diego Architecture
- Application SSH Components and Processes
- How the Diego Auction Allocates Jobs

Cloud Foundry Overview

Page last updated:

The Industry-Standard Cloud Platform

Cloud platforms let anyone deploy network apps or services and make them available to the world in a few minutes. When an app becomes popular, the cloud easily scales it to handle more traffic, replacing with a few keystrokes the build-out and migration efforts that once took months. Cloud platforms represent the next step in the evolution of IT, enabling you to focus exclusively on your applications and data without worrying about underlying infrastructure.



Not all cloud platforms are created equal. Some have limited language and framework support, lack key app services, or restrict deployment to a single cloud. Cloud Foundry (CF) has become the industry standard. It is an open source platform that you can deploy to run your apps on your own computing infrastructure, or deploy on an IaaS like AWS, vSphere, or OpenStack. You can also use a PaaS deployed by a commercial CF cloud provider. A broad community contributes to and supports Cloud Foundry. The platform's openness and extensibility prevent its users from being locked into a single framework, set of app services, or cloud.

Cloud Foundry is ideal for anyone interested in removing the cost and complexity of configuring infrastructure for their apps. Developers can deploy their apps to Cloud Foundry using their existing tools and with zero modification to their code.

How Cloud Foundry Works

To flexibly serve and scale apps online, Cloud Foundry has subsystems that perform specialized functions. Here's how some of these main subsystems work.

How the Cloud Balances Its Load

Clouds balance their processing loads over multiple machines, optimizing for efficiency and resilience against point failure. A Cloud Foundry installation accomplishes this at three levels:

1. BOSH creates and deploys virtual machines (VMs) on top of a physical computing infrastructure, and deploys and runs Cloud Foundry on top of this cloud. To configure the deployment, BOSH follows a manifest document.

2. The CF Cloud Controller runs the apps and other processes on the cloud's VMs, balancing demand and managing app lifecycles.

3. The router routes incoming traffic from the world to the VMs that are running the apps that the traffic demands, usually working with a customer-provided load balancer.

How Apps Run Anywhere

Cloud Foundry designates two types of VMs: the component VMs that constitute the platform's infrastructure, and the host VMs that host apps for the outside world. Within CF, the Diego system distributes the hosted app load over all of the host VMs, and keeps it running and balanced through demand surges, outages, or other changes. Diego accomplishes this through an auction algorithm.

To meet demand, multiple host VMs run duplicate instances of the same app. This means that apps must be portable. Cloud Foundry distributes app source code to VMs with everything the VMs need to compile and run the apps locally. This includes the OS stack that the app runs on, and a buildpack containing all languages, libraries, and services that the app uses. Before sending an app to a VM, the Cloud Controller stages it for delivery by combining stack, buildpack, and source code into a droplet that the VM can unpack, compile, and run. For simple, standalone apps with no dynamic pointers, the droplet can contain a pre-compiled executable instead of source code, language, and libraries.

How CF Organizes Users and Workspaces

CF manages user accounts through two User Authentication and Authorization (UAA) servers, which support access control as OAuth2 services and can store user information internally, or connect to external user stores through LDAP or SAML.

One UAA server grants access to BOSH, and holds accounts for the CF operators who deploy runtimes, services, and other software onto the BOSH layer directly. The other UAA server controls access to the Cloud Controller, and determines who can tell it to do what. The Cloud Controller UAA defines different user roles, such as admin, developer, or auditor, and grants them different sets of privileges to run CF commands. The Cloud Controller UAA also scopes the roles to separate, compartmentalized Orgs and Spaces within an installation, to manage and track use.

Where CF Stores Resources

Cloud Foundry uses the git system on GitHub to version-control source code, buildpacks, documentation, and other resources. Developers on the platform also use GitHub for their own apps, custom configurations, and other resources. To store large binary files, such as droplets, CF maintains an internal or external blobstore. To store and share temporary information, such as internal component states, CF uses MySQL and Consul.

How CF Components Communicate

Cloud Foundry components communicate with each other by posting messages internally using http and https protocols, and by sending NATS messages to each other directly.

How to Monitor and Analyze a CF Deployment

Pivotal Cloud Foundry (PCF) generates two types of logs, system logs from PCF components and app logs from hosted apps

As Cloud Foundry runs, its component and host VMs continuously generate logs and metrics, and Cloud Foundry apps also typically generate logs. The Loggregator system aggregates the component metrics and app logs into a structured, usable form, the **Firehose**. You can use all of the output of the Firehose, or direct the output to specific uses, such as monitoring system internals, triggering alerts, or analyzing user behavior, by applying **nozzles**.

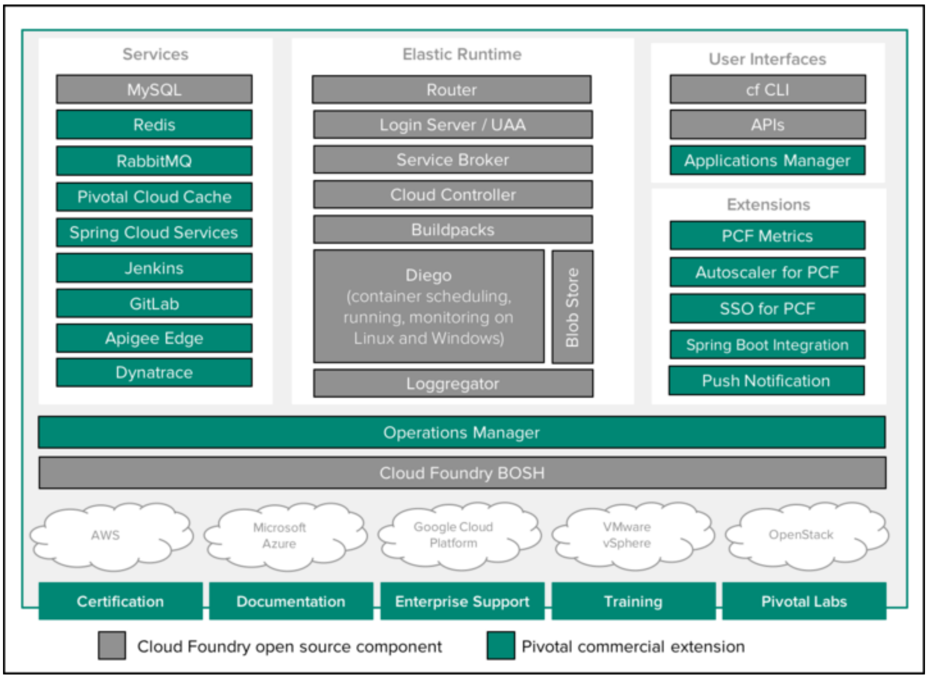
The component logs follow a different path. They stream from rsyslog agents, and the cloud operator can configure them to stream out to a syslog drain.

Using Services with CF

Typical apps depend on free or metered services such as databases or third-party APIs. To incorporate these into an app, a developer writes a Service Broker, an API that publishes to the Cloud Controller the ability to list service offerings, provision the service, and enable apps to make calls out to it.

How Pivotal Cloud Foundry Differs from Open Source Cloud Foundry

Open source software provides the basis for the Pivotal Cloud Foundry platform. Pivotal Application Service (PAS) is the Pivotal distribution of Cloud Foundry software for hosting apps. Pivotal offers additional commercial features, enterprise services, support, docs, certificates, and others value-adds.


Services
MySQL
Redis
RabbitMQ
Pivotal Cloud Cache
Spring Cloud Services
Jenkins
GitLab
Apigee Edge
Dynatrace
Elastic Runtime
Router
Login Server / UAA
Service Broker
Cloud Controller
Buildpacks
Diego
(container scheduling, running, monitoring on Linux and Windows)
Blob Store
Loggregator
User Interfaces
cf CLI
APIs
Applications Manager
Extensions
PCF Metrics
Autoscaler for PCF
SSO for PCF
Spring Boot Integration
Push Notification
Operations Manager
Cloud Foundry BOSH
AWS
Microsoft Azure
Google Cloud Platform
VMware vSphere
OpenStack
Certification
Documentation
Enterprise Support
Training
Pivotal Labs
Cloud Foundry open source component
Pivotal commercial extension

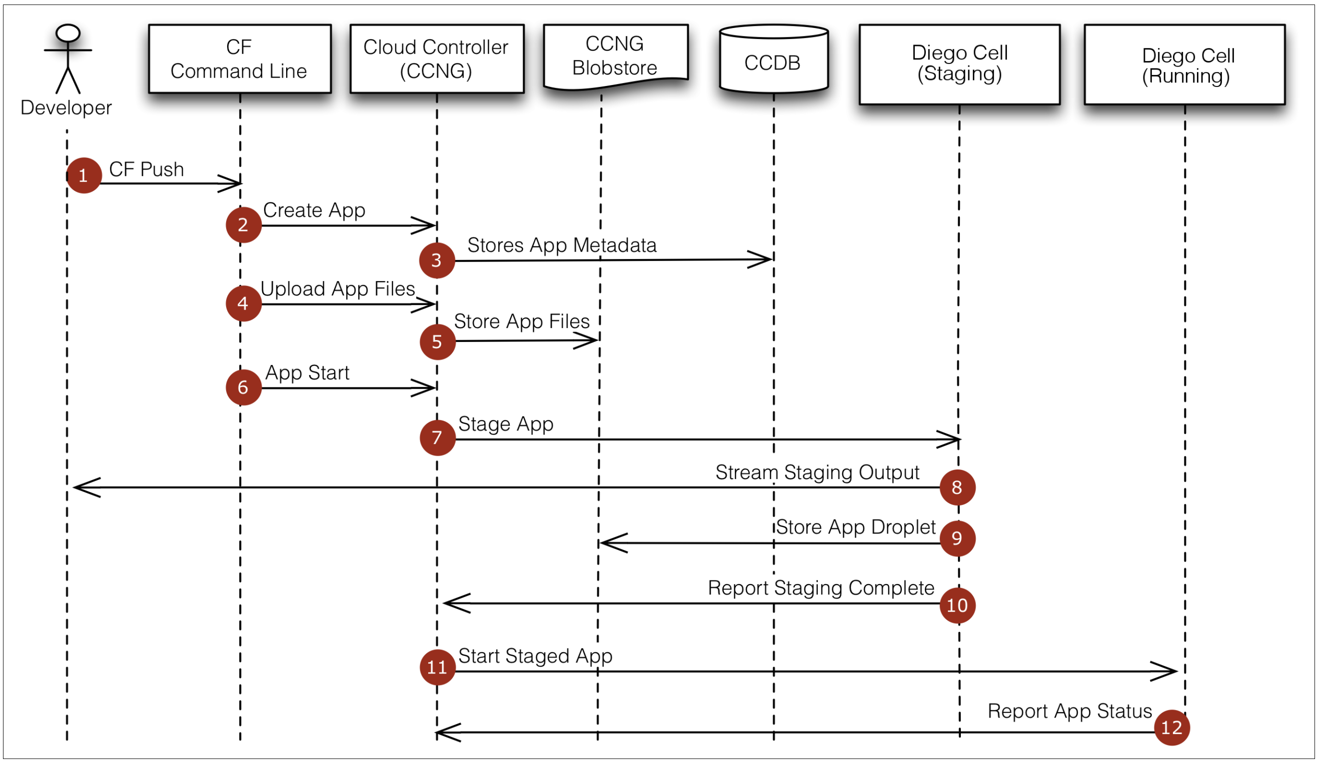
How Applications Are Staged

Page last updated:

Cloud Foundry uses Diego to manage application containers.

This topic describes how the Diego architecture stages buildpack applications and Docker images.

How Diego Stages Buildpack Applications



1. At the command line, the developer enters the directory containing her application source code and uses the Cloud Foundry Command Line Interface (cf CLI) to issue a push command.

2. The cf CLI tells the Cloud Controller to create a record for the application.

3. The Cloud Controller stores the application metadata. Application metadata can include the app name, number of instances the user specified, and the buildpack, and other information about the application.

4. Before uploading all the application source files, the cf CLI issues a resource match request to the Cloud Controller to determine if any of the application files already exist in the resource cache. When the application files are uploaded, the cf CLI omits files that exist in the resource cache by supplying the result of the resource match request. The uploaded application files are combined with the files from the resource cache to create the application package.

5. The Cloud Controller stores the application package in the blobstore.

6. The cf CLI issues an app start command.

7. The Cloud Controller issues a staging request to Diego, which then schedules a Diego cell ("Cell") to run the staging task ("Task"). The Task downloads buildpacks and the app's buildpack cache, if present. It then uses the buildpack that is detected automatically or specified with the `-b` flag to compile and stage the application.

8. The Cell streams the output of the staging process so the developer can troubleshoot application staging problems.

9. The Task packages the resulting compiled and staged application into a tarball called a "droplet" and the Cell stores the droplet in the blobstore. The Task also uploads the buildpack cache to the blobstore for use the next time the application is staged.

10. The Diego Bulletin Board System reports to the Cloud Controller that staging is complete. Staging must complete within 15 minutes or the staging is

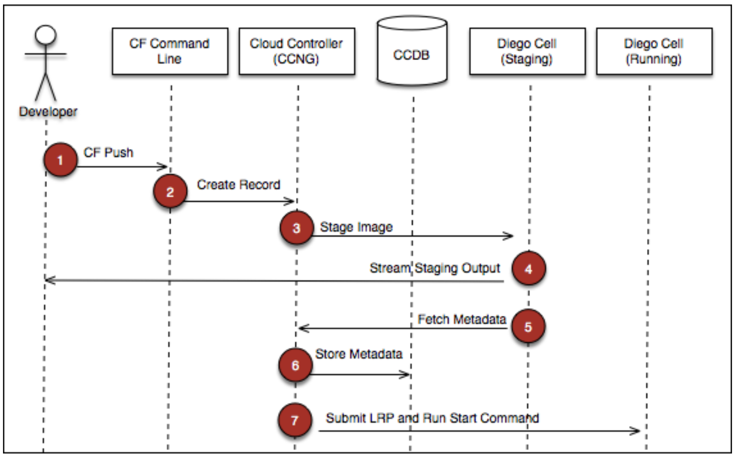
considered failed. Apps are given a minimum of 1GB memory to stage, even if the requested running memory is smaller.

11. Diego schedules the application as a Long Running Process on one or more Diego cells.

12. The Diego cells report the status of the application to the Cloud Controller.

See the Diego Architecture topic for more information.

How Diego Stages Docker Images



1. At the command line, the developer enters the name of a Docker image in an accessible Docker Registry and uses the cf CLI to issue a push command.

2. The cf CLI tells the Cloud Controller to create a record for the Docker image.

3. The Cloud Controller issues a staging request to Diego, which then schedules a Cell to run the Task.

4. The Cell streams the output of the staging process so the developer can troubleshoot staging problems.

5. The Task fetches the metadata associated with the Docker image and returns a portion of it to the Cloud Controller, which stores it in the Cloud Controller database (CCDB).

6. The Cloud Controller uses the Docker image metadata to construct a Long Running Process that runs the start command specified in the Dockerfile. The Cloud Controller also takes into account any user-specified overrides specified in the Dockerfile, such as custom environment variables.

7. The Cloud Controller submits the Long Running Process to Diego. Diego schedules the Long Running Process on one or more Diego Cells.

8. The Cloud Controller instructs Diego and the Gorouter to route traffic to the Docker image.

High Availability in Cloud Foundry

Page last updated:

This topic explains how to configure Cloud Foundry (CF) for high availability (HA), and how Cloud Foundry is designed to ensure high availability at multiple layers.

Note: In PCF v1.11 and later, PAS defaults to a highly available resource configuration. However, you may need to complete additional procedures described below to make your deployment highly available.

Configuring High Availability

This section describes how to configure system components to ensure high availability. You accomplish this by scaling component VMs and locating them in multiple Availability Zones (AZs), so that their redundancy and distribution minimizes downtime during ongoing operation, product updates, and platform upgrades.

Scaling component VMs means changing the number of VM instances dedicated to running a functional component of the system. Scaling usually means increasing this number, while scaling down or scaling back means decreasing it.

Deploying or scaling applications to at least two instances per app also helps maintain high availability. For information about scaling applications and maintaining app uptime, see the Scaling an Application Using cf scale and Using Blue-Green Deployment to Reduce Downtime and Risk topics.

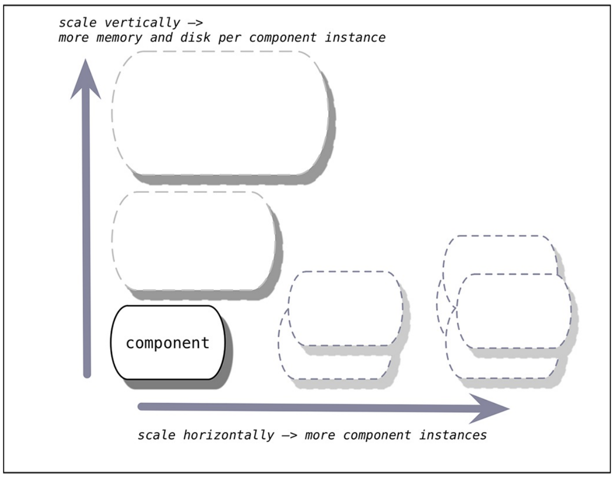
Availability Zones

During product updates and platform upgrades, the VMs in a deployment restart in succession, rendering them temporarily unavailable. During outages, VMs go down in a less orderly way. Spreading components across Availability Zones and scaling them to a sufficient level of redundancy maintains high availability during both upgrades and outages and can ensure zero downtime.

Deploying Cloud Foundry across three or more AZs and assigning multiple component instances to different AZ locations lets a deployment operate uninterrupted when entire AZs become unavailable. Cloud Foundry maintains its availability as long as a majority of the AZs remain accessible. For example, a three-AZ deployment stays up when one entire AZ goes down, and a five-AZ deployment can withstand an outage of up to two AZs with no impact on uptime.

Vertical and Horizontal Scaling

You can scale platform capacity vertically by adding memory and disk, or horizontally by adding more VMs running instances of Cloud Foundry components.



To scale vertically, ensure that you allocate and maintain enough of the following:

- Free space on host Diego cell VMs so that apps expected to deploy can successfully be staged and run.
- Disk space and memory in your deployment such that if one host VM is down, all instances of apps can be placed on the remaining Host VMs.

- Free space to handle one AZ going down if deploying in multiple AZs.

Scaling up the following components horizontally also increases your capacity to host applications. The nature of the applications you host on Cloud Foundry should determine how you should scale vertically vs. horizontally.

Scalable Components

You can horizontally scale most Cloud Foundry components to multiple instances to achieve the redundancy required for high availability.

You should also distribute the instances of multiply-scaled components across different availability zones (AZs). If you use more than three AZs, ensure that you use an odd number of AZs.

If you do not require a high-availability deployment (for example, you are deploying a proof-of-concept or sandbox), you should scale down job instances to the minimum required for a functional deployment.

For more information regarding zero downtime deployment, see the Scaling Instances in PAS topic.

The following table provides recommended instance counts for a high-availability deployment and the minimum instances for a functional deployment:

Pivotal Application Service (PAS) Job	Recommended Instance Number for HA	Minimum Instance Number	Notes
Diego Cell	≥ 3	1	The optimal balance between CPU/memory sizing and instance count depends on the performance characteristics of the apps that run on Diego cells. Scaling vertically with larger Diego cells makes for larger points of failure, and more apps go down when a cell fails. On the other hand, scaling horizontally decreases the speed at which the system rebalances apps. Rebalancing 100 cells takes longer and demands more processing overhead than rebalancing 20 cells.
Diego Brain	≥ 2	1	For high availability, use at least one per AZ, or at least two if only one AZ.
Diego BBS	≥ 3	1	Set this to an odd number equal to or one greater than the number of AZs you have, in order to maintain quorum. Distribute the instances evenly across the AZs, at least one instance per AZ.
Consul	≥ 3	1	Set this to an odd number equal to or one greater than the number of AZs you have, in order to maintain quorum. Distribute the instances evenly across the AZs, at least one instance per AZ.
MySQL Server	3	1	If you use an external database in your deployment, then you can set the MySQL Server instance count to `0`. For instructions about scaling down an internal MySQL cluster, see Scaling Down Your MySQL Cluster.
MySQL Proxy	2	1	If you use an external database in your deployment, then you can set the MySQL Proxy instance count to `0`.
NATS Server	≥ 2	1	In a high availability deployment, you might run a single NATS instance if your deployment lacks the resources to deploy two stable NATS servers. Components using NATS are resilient to message failures and the BOSH resurrector recovers the NATS VM quickly if it becomes non-responsive.
Cloud Controller	≥ 2	1	Scale the Cloud Controller to accommodate the number of requests to the API and the number of apps in the system.
Clock Global	≥ 2	1	For a high availability deployment, scale the Clock Global job to a value greater than 1 or to the number of AZs you have.
Router	≥ 2	1	Scale the router to accommodate the number of incoming requests. Additional instances increase available bandwidth. In general, this load is much less than the load on Diego cells.
HAProxy	0 or ≥ 2	0 or 1	For environments that require high availability, you can scale HAProxy to `0` and then configure a high-availability load balancer (LB) to point directly to each Gorouter instance. Alternately, you can also configure the high availability LB to point to HAProxy instance scaled at `≥ 2`. Either way, an LB is required to host Cloud Foundry domains at a single IP address.
UAA	≥ 2	1	
Doppler Server	≥ 2	1	Deploying additional Doppler servers splits traffic across them. For a high availability deployment, Pivotal recommends at least two per Availability Zone.
Loggregator Trafficcontroller	≥ 2	1	Deploying additional Loggregator Traffic Controllers allows you to direct traffic to them in a round-robin manner. For a high availability deployment, Pivotal recommends at least two per Availability Zone.
Syslog Scheduler	≥ 2	1	The Syslog Scheduler is a scalable component. For high availability, use at least one instance per AZ, or at least two instances if only one AZ is present.

Blob Storage

For storing blobs, large binary files, the best approach for high availability is to use external storage such as Amazon S3 or an S3-compatible service.

If you store blobs internally using WebDAV or NFS, these components run as single instances and you cannot scale them. For these deployments, use the high availability features of your IaaS to immediately recover your WebDAV or NFS server VM if it fails. Contact Pivotal Support if you need assistance.

The singleton compilation components do not affect platform availability.

Supporting Component Scaling

Ops Manager Resurrector

Enable the Ops Manager Resurrector.

Resource Pools

Configure your resource pools according to the requirements of your deployment.

Each IaaS has different ways of limiting resource consumption for scaling VMs. Consult with your IaaS administrator to ensure additional VMs and related resources, like IPs and storage, will be available when scaling.

For Amazon Web Services, review the documentation regarding scaling instances. If you are using OpenStack, see the topic regarding managing projects and users. For vSphere, review the Configuring Ops Manager Director for VMware vSphere topic.

Databases

For database services deployed outside Cloud Foundry, plan to leverage your infrastructure's high availability features and to configure backup and restore where possible. For more information about scaling internal database components, see the Scaling Instances in PAS topic.

Note: Data services may have single points of failure depending on their configuration.

Contact Pivotal Support if you need assistance.

How CF Maintains High Availability

This section explains how Pivotal Cloud Foundry (PCF) deployments include several layers of HA to keep applications running in the face of system failure. These layers include availability zones (AZs), application health management, process monitoring, and VM resurrection.

Availability Zones

PCF supports deploying applications instances across multiple AZs. This level of high availability requires that you define AZs in your IaaS. PCF balances the applications you deploy across the AZs you defined. If an AZ goes down, you still have application instances running in another.

You can configure your deployment so that Diego cells are created across these AZs. Follow the configuration for your specific IaaS: AWS, GCP, OpenStack, or vSphere.

Health Management for App Instances

If you lose application instances for any reason, such as a bug in the app or an AZ going down, PCF restarts new instances to maintain capacity. Under Diego architecture, the nsync, BBS, and Cell Rep components track the number of instances of each application that are running across all of the Diego cells. When these components detect a discrepancy between the actual state of the app instances in the cloud and the desired state as known by the Cloud Controller, they advise the Cloud Controller of the difference and the Cloud Controller initiates the deployment of new application instances.

Process Monitoring

PCF uses a BOSH agent, monit, to monitor the processes on the component VMs that work together to keep your applications running, such as nsync, BBS, and Cell Rep. If monit detects a failure, it restarts the process and notifies the BOSH agent on the VM. The BOSH agent notifies the BOSH Health Monitor, which triggers responders through plugins such as email notifications or paging.

Resurrection for VMs

BOSH detects if a VM is present by listening for heartbeat messages that are sent from the BOSH agent every 60 seconds. The BOSH Health Monitor listens for those heartbeats. When the Health Monitor finds that a VM is not responding, it passes an alert to the Resurrector component. If the Resurrector is enabled, it sends the IaaS a request to create a new VM instance to replace the one that failed.

To enable the Resurrector, see the following pages for your particular IaaS: AWS, Azure, GCP, OpenStack, or vSphere.

Orgs, Spaces, Roles, and Permissions

PCF uses a role-based access control (RBAC) system to grant Pivotal Application Service users permissions appropriate to their role within an org or a space. This topic describes how orgs and spaces work within a PCF deployment, and how different Pivotal Application Service User roles operate within those contexts.

Admins, Org Managers, and Space Managers can assign user roles using the cf CLI or Apps Manager.

Note: Before you assign a **space role** to a user, you must assign an **org role** to the user.

Orgs

An org is a development account that an individual or multiple collaborators can own and use. All collaborators access an org with user accounts. Collaborators in an org share a resource quota plan, applications, services availability, and custom domains.

By default, an org has the status of *active*. An admin can set the status of an org to *suspended* for various reasons such as failure to provide payment or misuse. When an org is suspended, users cannot perform certain activities within the org, such as push apps, modify spaces, or bind services. For details on what activities are allowed for suspended orgs, see Roles and Permissions for Suspended Orgs.

User Accounts

A user account represents an individual person within the context of a PCF installation. A user can have different roles in different spaces within an org, governing what level and type of access they have within that space.

Before you assign a space role to a user, you must assign an org role to the user. The error message `Server error, error code: 1002, message: cannot set space role because user is not part of the org` occurs when you try to set a space role before setting an org role for the user.

Spaces

Every application and service is scoped to a space. Each org contains at least one space. A space provides users with access to a shared location for application development, deployment, and maintenance. Each space role applies only to a particular space.

Roles and Permissions

A user can have one or more roles. The combination of these roles defines the user's overall permissions in the org and within specific spaces in that org.

For non-admin users, the `cloud_controller.read` scope is required to view resources, and the `cloud_controller.write` scope is required to create, update, and delete resources.

- **Admin** is a user role that has been assigned the `cloud_controller.admin` scope in UAA. An admin user has permissions on all orgs and spaces and can perform operational actions using the Cloud Controller API. To create an account with `cloud_controller.admin` scope for your installation, see Create an Admin User.
- **Admin Read-Only** is a user role that has been assigned the `cloud_controller.admin_read_only` scope in UAA. This role has read-only access to all Cloud Controller API resources.
- **Global Auditor** is a user role that has been assigned the `cloud_controller.global_auditor` scope in UAA. This role has read-only access to all Cloud Controller API resources except for secrets such as environment variables. The Global Auditor role cannot access those values.
- **Org Managers** are managers or other users who need to administer the org.
- **Org Auditors** view but cannot edit user information and org quota usage information.
- **Org Users** can view the list of other org users and their roles. When an Org Manager gives a person an Org or Space role, that person automatically receives Org User status in that Org.
- **Space Managers** are managers or other users who administer a space within an org.
- **Space Developers** are application developers or other users who manage applications and services in a space.

- **Space Auditors** view but cannot edit the space.

Roles and Permissions for Active Orgs

The following table describes the permissions for various PCF roles.

Activity	Admin	Admin Read-Only	Global Auditor	Org Manager	Org Auditor	Space Manager	Space Developer	Space Auditor
Scope of operation	Org	Org	Org	Org	Org	Space	Space	Space
Add and edit users and roles	✓			**		**		
View users and roles	✓	✓	✓	✓	✓	✓	✓	✓
Create and assign org quota plans	✓							
View org quota plans	✓	✓	✓	✓	✓	✓	✓	✓
Create orgs	✓			*	*	*	*	*
View all orgs	✓	✓	✓					
View orgs where user is a member	✓***	✓***	✓***	✓	✓	✓	✓	✓
Edit, rename, and delete orgs	✓							
Suspend or activate an org	✓							
Create and assign space quota plans	✓			✓				
Create spaces	✓			✓				
View spaces	✓	✓	✓	✓		✓	✓	✓
Edit spaces	✓			✓		✓		
Delete spaces	✓			✓				
Rename spaces	✓			✓		✓		
View the status, number of instances, service bindings, and resource use of applications	✓	✓	✓	✓		✓	✓	✓
Add private domains†	✓			✓				
Deploy, run, and manage applications	✓						✓	
Instantiate and bind services to applications	✓						✓	
Associate routes†, instance counts, memory allocation, and disk limit of applications	✓						✓	
Rename applications	✓						✓	
Create and manage Application Security Groups	✓							
Create, update, and delete an Isolation Segment	✓							
List all Isolation Segments for an Org	✓	✓	✓****	✓****	✓****	✓****	✓****	✓****
Entitle or revoke an Isolation Segment	✓							
List all Orgs entitled to an Isolation Segment	✓	✓	✓****	✓****	✓****	✓****	✓****	✓****
Assign a default Isolation Segment to an Org	✓			✓				
List and manage Isolation Segments for spaces	✓			✓				
List entitled Isolation Segment for a space	✓	✓	✓	✓		✓	✓	✓
List which Isolation Segment an app runs on	✓	✓	✓	✓		✓	✓	✓

*Not by default, unless feature flag `user_org_creation` is set to `true`.

**Not by default, unless feature flag `set_roles_by_username` is set to `true`.

***Admin, admin read-only, and global auditor roles do not need to be added as members of orgs or spaces to view resources.

****Applies only to orgs they belong to.

†Unless disabled by feature flags.

Roles and Permissions for Suspended Orgs

The following table describes roles and permissions applied after an operator sets the status of an org to*suspended*.

Activity	Admin	Admin Read-Only	Global Auditor	Org Manager	Org Auditor	Space Manager	Space Developer	Space Auditor
Scope of operation	Org	Org	Org	Org	Org	Space	Space	Space
Add and edit users and roles	✓							
View users and roles	✓	✓	✓	✓	✓	✓	✓	✓
Create and assign org quota plans	✓							
View org quota plans	✓	✓	✓	✓	✓	✓	✓	✓
Create orgs	✓							
View all orgs	✓	✓	✓					
View orgs where user is a member	✓	✓	✓	✓	✓	✓	✓	✓
Edit, rename, and delete orgs	✓							
Suspend or activate an org	✓							
Create and assign space quota plans	✓							
Create spaces	✓							
View spaces	✓	✓	✓	✓		✓	✓	✓
Edit spaces	✓							
Delete spaces	✓							
Rename spaces	✓							
View the status, number of instances, service bindings, and resource use of applications	✓	✓	✓	✓		✓	✓	✓
Add private domains†	✓							
Deploy, run, and manage applications	✓							
Instantiate and bind services to applications	✓							
Associate routes†, instance counts, memory allocation, and disk limit of applications	✓							
Rename applications	✓							
Create and manage Application Security Groups	✓							

Understanding Cloud Foundry Security

Page last updated:

This topic provides an overview of Cloud Foundry (CF) security. For an overview of container security, see the Understanding Container Security topic.

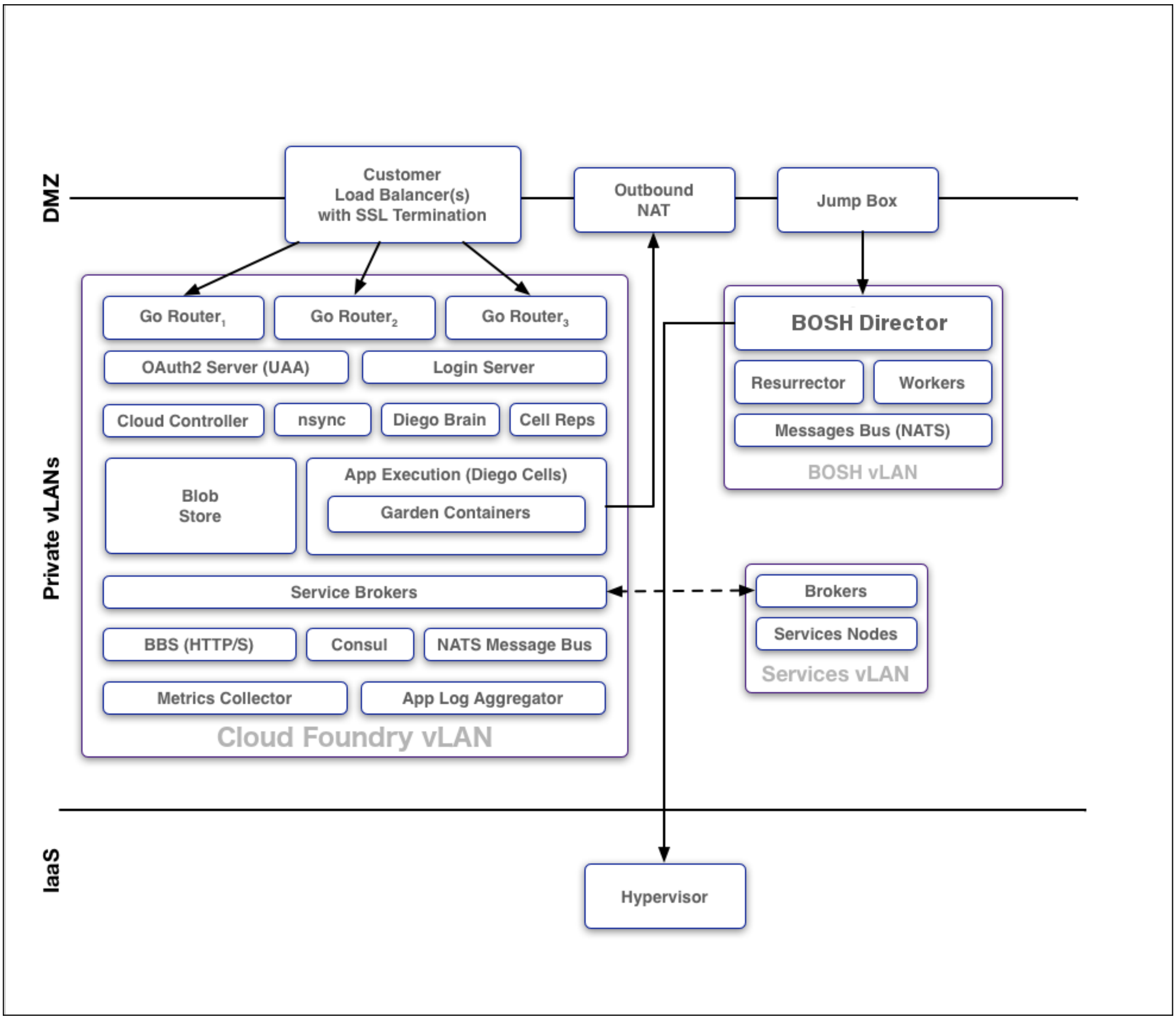
Cloud Foundry implements the following measures to mitigate against security threats:

- Minimizes network surface area
- Isolates customer applications and data in containers
- Encrypts connections
- Uses role-based access controls, applying and enforcing roles and permissions to ensure that users can only view and affect the spaces for which they have been granted access
- Ensures security of application bits in a multi-tenant environment
- Prevents possible denial of service attacks through resource starvation

System Boundaries and Access

As the image below shows, in a typical deployment of Cloud Foundry, the components run on virtual machines (VMs) that exist within a VLAN. In this configuration, the only access points visible on a public network are a load balancer that maps to one or more Cloud Foundry routers and, optionally, a NAT VM and a jumpbox. Because of the limited number of contact points with the public internet, the surface area for possible security vulnerabilities is minimized.

Note: Pivotal recommends that you also install a NAT VM for outbound requests and a Jumpbox to access the BOSH Director, though these access points are optional depending on your network configuration.



Protocols

All traffic from the public internet to the Cloud Controller and UAA happens over HTTPS. Inside the boundary of the system, components communicate over a publish-subscribe (pub-sub) message bus NATS, HTTP, and SSL/TLS.

BOSH

Operators deploy Cloud Foundry with BOSH. The BOSH Director is the core orchestrating component in BOSH: it controls VM creation and deployment, as well as other software and service lifecycle events. You use HTTPS to ensure secure communication to the BOSH Director.

> **Note**: Pivotal recommends that you deploy the BOSH Director on a subnet that is not publicly accessible, and access the BOSH Director from a Jumpbox on the subnet or through VPN.

BOSH includes the following functionality for security:

- Communicates with the VMs it launches over NATS. Because NATS cannot be accessed from outside Cloud Foundry, this ensures that published messages can only originate from a component within your deployment.
- Provides an audit trail through the `bosh tasks --all` and `bosh tasks --recent=VALUE` commands. `bosh tasks --all` returns a table that shows all BOSH actions taken by an operator or other running processes. `bosh tasks --recent=VALUE` returns a table of recent tasks, with `VALUE` being the number of recent tasks you want to view.
- Allows you to set up individual login accounts for each operator. BOSH operators have root access.

> **Note**: BOSH does not encrypt data stored on BOSH VMs. Your IaaS might encrypt this data.

Isolation Segments

Isolation segments provide dedicated pools of resources to which apps can be deployed to isolate workloads. Using isolation segments separates app resources as completely as if they were in different CF deployments but avoids redundant management components and unneeded network complexity.

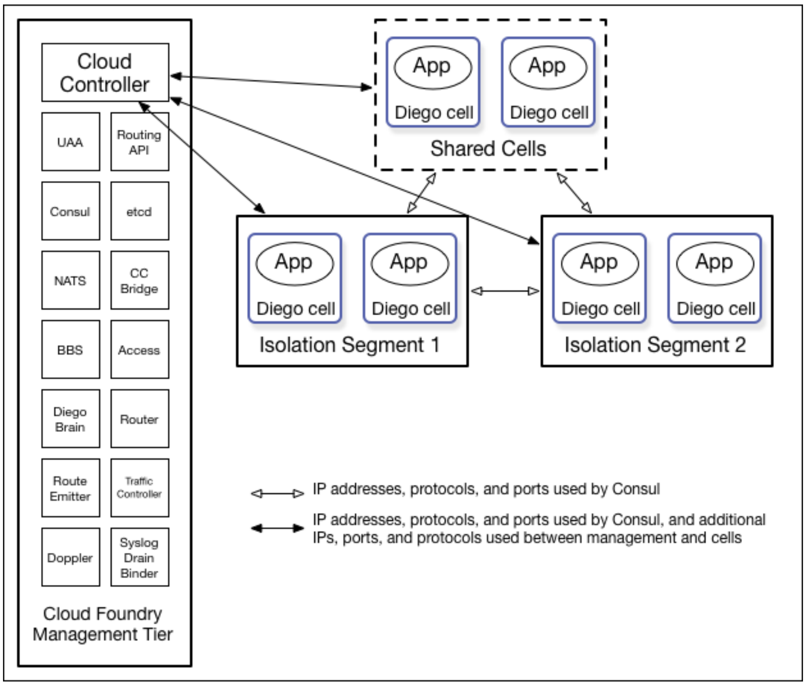
You can designate isolation segments for exclusive use by orgs and spaces within CF. This guarantees that apps within the org or space use resources that are not also used by other orgs or spaces.

Customers can use isolation segments for different reasons, including the following:

- To follow regulatory restrictions that require separation between different types of applications. For example, a health care company may not be able to host medical records and billing systems on the same machines.
- To dedicate specific hardware to different isolation segments. For example, to guarantee that high-priority apps run on a cluster of high-performance hosts.
- To separate data on multiple clients, to strengthen a security story, or offer different hosting tiers.

In CF, the Cloud Controller Database (CCDB) identifies isolation segments by name and GUID, for example `30dd879c-ee2f-11db-8314-0800200c9a66`. The isolation segment object has no internal structure beyond these two properties at the Cloud Foundry level, but BOSH associates the name of the isolation segment with Diego cells, through their `placement_tag` property.

This diagram shows how isolation segments keep apps running on different pools of cells, and how the cells communicate with each other and with the management components:



See the Installing PCF Isolation Segment and Managing Isolation Segments topics for more information about how to create and manage isolation segments in a PCF deployment.

See the Isolation Segments section of the *Cloud Controller API (CAPI) Reference* for API commands related to isolation segments.

Authentication and Authorization

User Account and Authentication (UAA) is the central identity management service for Cloud Foundry and its various components.

UAA acts as an OAuth2 Authorization Server and issues access tokens for applications that request platform resources. The tokens are based on the JSON Web Token and are digitally signed by UAA.

Operators can configure the identity store in UAA. If users register an account with the Cloud Foundry platform, UAA acts as the user store and stores user passwords in the UAA database using bcrypt. UAA also supports connecting to external user stores through LDAP and SAML. Once an operator has configured the external user store, such as a corporate Microsoft Active Directory, users can use their LDAP credentials to gain access to the Cloud Foundry platform instead of registering a separate account. Alternatively, operators can use SAML to connect to an external user store and enable single sign-on for users into the Cloud Foundry platform.

Managing User Access with Role-Based Access Control

Applications that users deploy to Cloud Foundry exist within a space. Spaces exist within orgs. To view and access an org or a space, a user must be a member of it. Cloud Foundry uses role-based access control (RBAC), with each role granted permissions to either an org or a specified space. For more information about roles and permissions, refer to the Orgs, Spaces, Roles, and Permissions topic.

For more information, see Getting Started with the Apps Manager and Managing User Accounts and Permissions Using the Apps Manager.

Security for Service Broker Integration

The Cloud Controller authenticates every request with the Service Broker API using HTTP or HTTPS, depending on which protocol that you specify during broker registration. The Cloud Controller rejects any broker registration that does not contain a username and password.

Service instances bound to an app contain credential data. Users specify the binding credentials for user-provided service instances, while third-party brokers specify the binding credentials for managed service instances. The VCAP_SERVICES environment variable contains credential information for any service bound to an app. Cloud Foundry constructs this value from encrypted data that it stores in the Cloud Controller Database (CCDB).

Note: The selected third-party broker controls how securely to communicate managed service credentials.

A third-party broker might offer a dashboard client in its catalog. Dashboard clients require a text string defined as a `client_secret`. Cloud Foundry does not store this secret in the CCDB. Instead, Cloud Foundry passes the secret to the UAA component for verification using HTTP or HTTPS.

Software Vulnerability Management

Cloud Foundry manages software vulnerability using releases and BOSH stemcells. New Cloud Foundry releases are created with updates to address code issues, while new stemcells are created with patches for the latest security fixes to address any underlying operating system issues.

Ensuring Security for Application Artifacts

Cloud Foundry secures both the code and the configuration of an application using the following functionality:

- Application developers push their code using the Cloud Foundry API. Cloud Foundry secures each call to the CF API using the UAA and SSL.
- The Cloud Controller uses RBAC to ensure that only authorized users can access a particular application.
- The Cloud Controller stores the configuration for an application in an encrypted database table. This configuration data includes user-specified environment variables and service credentials for any services bound to the app.
- Cloud Foundry runs the app inside a secure container. For more information, see the Understanding Container Security topic.
- Cloud Foundry operators can configure network traffic rules to control inbound communication to and outbound communication from an app. For more information, see the Network Traffic Rules section of the Understanding Container Security topic.

Security Event Logging and Auditing

For operators, Cloud Foundry provides an audit trail through the `bosh tasks` command. This command shows all actions that an operator has taken with the platform. Additionally, operators can redirect Cloud Foundry component logs to a standard syslog server using the `syslog_daemon_config` property in the `metron_agent` job of `cf-release`.

For users, Cloud Foundry records an audit trail of all relevant API invocations of an app. The Cloud Foundry Command Line Interface (cf CLI) command `cf events` returns this information.

Recommendations for Running a Secure Deployment

To help run a secure deployment, Pivotal recommends the following:

- Configure UAA clients and users using a BOSH manifest. Limit and manage these clients and users as you would any other kind of privileged account.
- Deploy within a VLAN that limits network traffic to individual VMs. This reduce the possibility of unauthorized access to the VMs within your BOSH-managed cloud.
- Enable HTTPS for applications and SSL database connections to protect sensitive data transmitted to and from applications.
- Ensure that the Jumpbox is secure, along with the load balancer and NAT VM.
- Encrypt stored files and data within databases to meet your data security requirements. Deploy using industry standard encryption and the best practices for your language or framework.
- Prohibit promiscuous network interfaces on the trusted network.
- Review and monitor data sharing and security practices with third-party services that you use to provide additional functionality to your application.
- Store SSH keys securely to prevent disclosure, and promptly replace lost or compromised keys.
- Use Cloud Foundry’s RBAC model to restrict your users’ access to only what is necessary to complete their tasks.
- Use a strong passphrase for both your Cloud Foundry user account and SSH keys.
- Use the IPsec add-on to encrypt IP data traffic within your deployment.

Understanding Container Security

Page last updated:

This topic describes how Cloud Foundry (CF) secures the containers that host application instances on Linux. For an overview of other CF security features, see the Understanding Cloud Foundry Security topic.

- Container Mechanics provides an overview of container isolation.
- Inbound and Outbound Traffic from CF provides an overview of container networking and describes how CF administrators customize container network traffic rules for their deployment.
- Container Security describes how CF secures containers by running application instances in unprivileged containers and by hardening them.

Container Mechanics

Each instance of an app deployed to CF runs within its own self-contained environment, a Garden container. This container isolates processes, memory, and the filesystem using operating system features and the characteristics of the virtual and physical infrastructure where CF is deployed.

CF achieves container isolation by namespacing kernel resources that would otherwise be shared. The intended level of isolation is set to prevent multiple containers that are present on the same host from detecting each other. Every container includes a private root filesystem, which includes a Process ID (PID), namespace, network namespace, and mount namespace.

CF creates container filesystems using the Garden Rootfs (GrootFS) tool. It stacks the following using OverlayFS:

- A **read-only base filesystem**: This filesystem has the minimal set of operating system packages and Garden-specific modifications common to all containers. Containers can share the same read-only base filesystem because all writes are applied to the read-write layer.
- A **container-specific read-write layer**: This layer is unique to each container and its size is limited by XFS project quotas. The quotas prevent the read-write layer from overflowing into unallocated space.

Resource control is managed using Linux control groups (cgroups). Associating each container with its own cgroup or job object limits the amount of memory that the container may use. Linux cgroups also require the container to use a fair share of CPU compared to the relative CPU share of other containers.

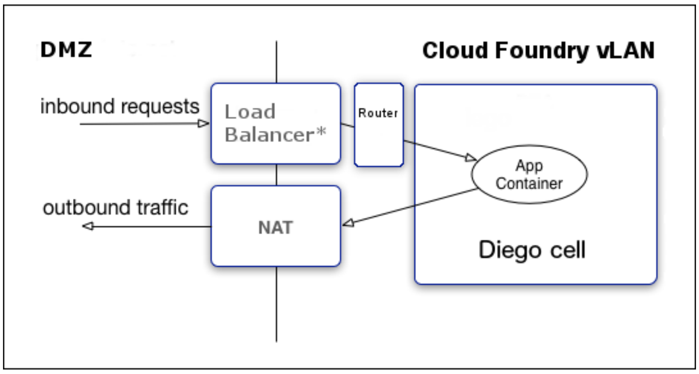
Note: CF does not support a RedHat Enterprise Linux OS stemcell. This is due to an inherent security issue with the way RedHat handles user namespacing and container isolation.

Inbound and Outbound Traffic from CF

Networking Overview

A host VM has a single IP address. If you configure the deployment with the cluster on a VLAN, as recommended, then all traffic goes through the following levels of network address translation, as shown in the diagram below.

- **Inbound** requests flow from the load balancer through the router to the host cell, then into the application container. The router determines which application instance receives each request.
- **Outbound** traffic flows from the application container to the cell, then to the gateway on the cell's virtual network interface. Depending on your IaaS, this gateway may be a NAT to external networks.



Network Traffic Rules

Administrators configure rules to govern container network traffic. This is how containers send traffic outside of CF and receive traffic from outside, the Internet. These rules can prevent system access from external networks and between internal components and determine if apps can establish connections over the virtual network interface.

Administrators configure these rules at two levels:

- Application Security Groups (ASGs) apply network traffic rules at the container level. For information about creating and configuring ASGs, see Application Security Groups.
- Container-to-Container networking policies determine app-to-app communication. Within CF, apps can communicate directly with each other, but the containers are isolated from outside CF. For information about administering container-to-container network policies, see Administering Container-to-Container Networking.

Container Security

CF secures containers through the following measures:

- Running application instances in unprivileged containers by default
- Hardening containers by limiting functionality and access rights
- Allowing administrators to configure ASGs to block outbound connections from application containers. For information about creating and configuring ASGs, see Application Security Groups.

Types

Garden has two container types: unprivileged and privileged. Currently, CF runs all application instances and staging tasks in unprivileged containers by default. This measure increases security by eliminating the threat of root escalation inside the container.

Formerly, CF ran apps based on Docker images in unprivileged containers, and buildpack-based apps and staging tasks in privileged containers. CF ran apps based on Docker images in unprivileged containers because Docker images come with their own root filesystem and user, so CF could not trust the root filesystem and could not assume that the container user process would never be root. CF ran build-pack based apps and staging tasks in privileged containers because they used the cflinuxfs2 root filesystem and all processes were run as the unprivileged user `vcap`.

Hardening

CF mitigates against container breakout and denial of service attacks in the following ways:

- CF uses the full set of Linux namespaces (IPC, Network, Mount, PID, User, UTS) to provide isolation between containers running on the same host. The User namespace is not used for privileged containers.
- In unprivileged containers, CF maps UID/GID 0 (root) inside the container user namespace to a different UID/GID on the host to prevent an app from inheriting UID/GID 0 on the host if it breaks out of the container.
  - CF uses the same UID/GID for all containers.
  - CF maps all UIDs except UID 0 to themselves. CF maps UID 0 inside the container namespace to `MAX_UID-1` outside of the container namespace.
  - Container Root does not grant Host Root permissions.

- CF mounts `/proc` and `/sys` as read-only inside containers.
- CF disallows `dmesg` access for unprivileged users and all users in unprivileged containers.
- CF uses `chroot` when importing docker images from docker registries.
- CF establishes a container-specific overlay filesystem mount. CF uses `pivot_root` to move the root filesystem into this overlay, in order to isolate the container from the host system's filesystem.
- CF does not call any binary or script inside the container filesystem, in order to eliminate any dependencies on scripts and binaries inside the root filesystem.
- CF avoids side-loading binaries in the container through bind mounts or other methods. Instead, it re-executes the same binary by reading it from `/proc/self/exe` whenever it needs to run a binary in a container.
- CF establishes a virtual Ethernet pair for each container for network traffic. See the Container Network Traffic section above for more information. The virtual Ethernet pair has the following features:
  - One interface in the pair is inside the container's network namespace, and is the only non-loopback interface accessible inside the container.
  - The other interface remains in the host network namespace and is bridged to the container-side interface.
  - Egress whitelist rules are applied to these interfaces according to Application Security Groups (ASGs) configured by the administrator.
  - First-packet logging rules may also be enabled on TCP whitelist rules.
  - DNAT rules are established on the host to enable traffic ingress from the host interface to whitelisted ports on the container-side interface.
- CF applies disk quotas using container-specific XFS quotas with the specified disk-quota capacity.
- CF applies a total memory usage quota through the memory cgroup and destroys the container if the memory usage exceeds the quota.
- CF applies a fair-use limit to CPU usage for processes inside the container through the `cpu.shares` control group.
- CF limits access to devices using cgroups but explicitly whitelists the following safe device nodes:
  - `/dev/full`
  - `/dev/fuse`
  - `/dev/null`
  - `/dev/ptmx`
  - `/dev/pts/*`
  - `/dev/random`
  - `/dev/tty`
  - `/dev/tty0`
  - `/dev/tty1`
  - `/dev/urandom`
  - `/dev/zero`
  - `/dev/tap`
  - `/dev/tun`
- CF drops the following Linux capabilities for all container processes. Every dropped capability limits what actions the root user can perform.
  - `CAP_DAC_READ_SEARCH`
  - `CAP_LINUX_IMMUTABLE`
  - `CAP_NET_BROADCAST`
  - `CAP_NET_ADMIN`
  - `CAP_IPC_LOCK`
  - `CAP_IPC_OWNER`
  - `CAP_SYS_MODULE`
  - `CAP_SYS_RAWIO`
  - `CAP_SYS_PTRACE`
  - `CAP_SYS_PACCT`
  - `CAP_SYS_BOOT`
  - `CAP_SYS_NICE`
  - `CAP_SYS_RESOURCE`
  - `CAP_SYS_TIME`
  - `CAP_SYS_TTY_CONFIG`
  - `CAP_LEASE`
  - `CAP_AUDIT_CONTROL`
  - `CAP_MAC_OVERRIDE`
  - `CAP_MAC_ADMIN`
  - `CAP_SYSLOG`
  - `CAP_WAKE_ALARM`
  - `CAP_BLOCK_SUSPEND`
  - `CAP_SYS_ADMIN` (for unprivileged containers)

Understanding Container-to-Container Networking

This topic provides an overview of how Container-to-Container Networking works.

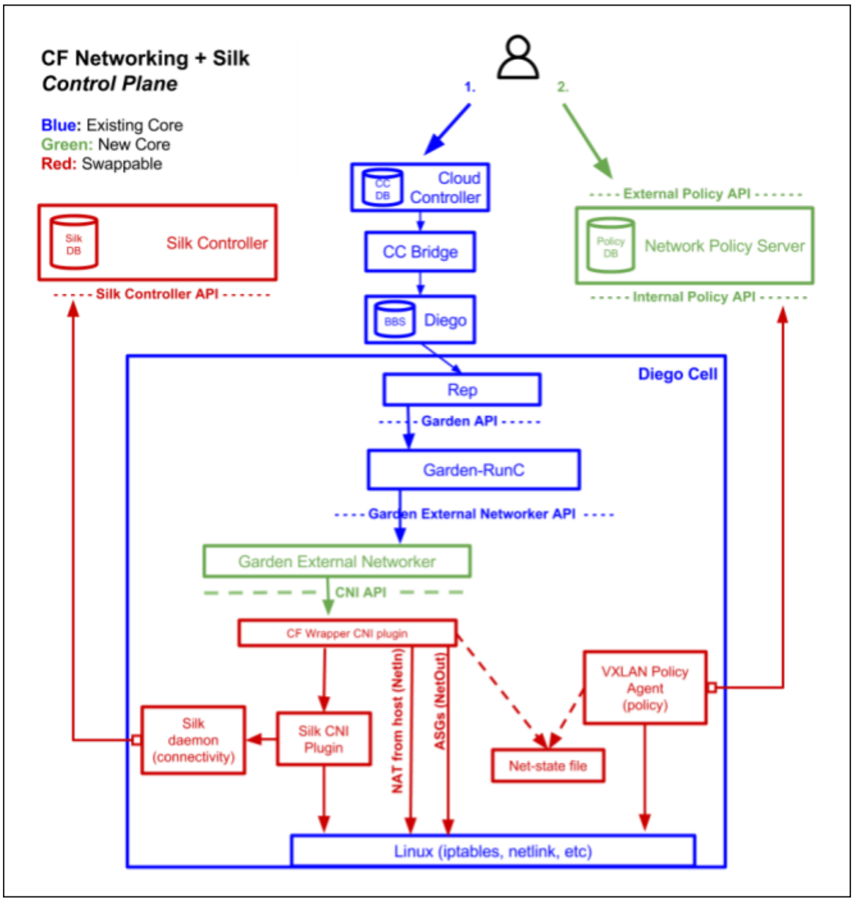
The Container-to-Container Networking feature enables app instances to communicate with each other directly. Container-to-Container Networking is always enabled in PAS. For more information about how to configure Container-to-Container Networking, see the Administering Container-to-Container Networking topic.

Architecture

Overview

Container-to-Container Networking integrates with Garden-runC in a Diego deployment. The Container-to-Container Networking BOSH release includes several core components, as well as swappable components.

To understand the components and how they work, see the diagram and tables below. The diagram highlights Pivotal Application Service components in blue and green. The diagram also highlights swappable components in red.



Core Components

The Container-to-Container Networking BOSH release includes the following core components:

Part	Function
Cloud Foundry Command Line Interface (CF CLI) plugin	A plugin that you download to control network access policies between apps.
	A central management node that does the following: • Maintains a database of policies for traffic between apps. The CF CLI plugin calls an API to create or update a

Policy Server	record in the policy database whenever you create or remove a policy. • Exposes a JSON REST API used by the CF CLI plugin
Garden External Networker	A Garden-runC add-on deployed to every Diego cell that does the following: • Invokes the CNI plugin component to set up the network for each app • Forwards ports to support incoming connections from the Gorouter, TCP Router, and Diego SSH Proxy. This keeps apps externally reachable. • Installs outbound whitelist rules to support Application Security Groups (ASGs).

Swappable Components

The Container-to-Container Networking BOSH release includes the following swappable components:

Part	Function
Silk CNI plugin	A plugin that provides IP address management and network connectivity to app instances as follows: • Uses a shared VXLAN overlay network to assign each container a unique IP address • Installs network interface in container using the Silk VXLAN backend. This is a shared, flat L3 network.
VXLAN Policy Agent	Enforces network policy for traffic between apps as follows: • Discovers desired network policies from the Policy Server Internal API • Updates iptables rules on the Diego cell to allow whitelisted inbound traffic • Tags outbound traffic with the unique identifier of the source app using the VXLAN Group-Based Policy (GBP) header

App Instance Communication

Container-to-Container Networking uses an overlay network to manage communication between app instances.

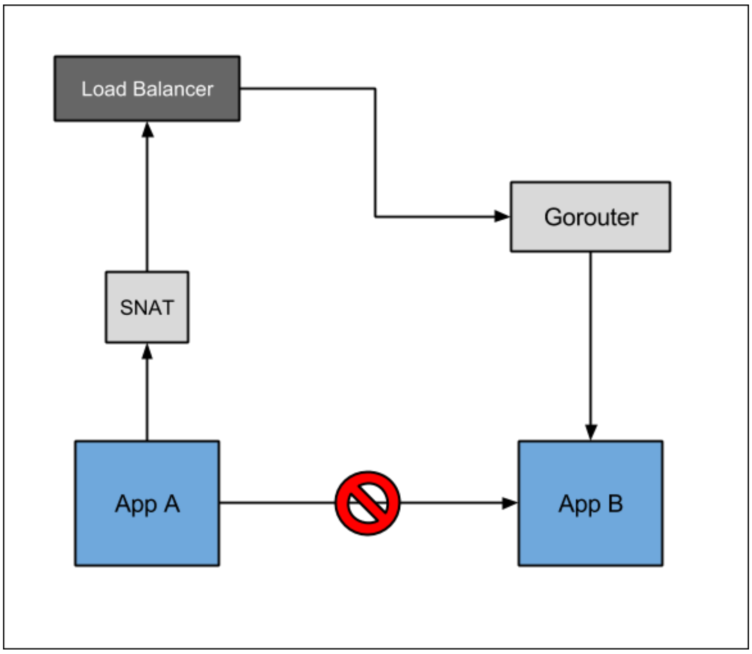
Each Diego cell assigns IP addresses to its containers using a /24 subnet range, which supports up to 250 containers per cell. The subnet range you choose for your overlay network determines how many Diego cells your network supports.

Using the /16 subnet range, Pivotal Application Service supports up to 250 Diego cells. To support a higher or lower number of Diego cells, choose a different subnet range. For example, use /12 to support more Diego cells or /24 to support fewer Diego cells.

> **NOTE**: The overlay network IP range must not conflict with any other IP addresses in the network. If there is a conflict, the Diego cells cannot reach any endpoint that has a conflicting IP address.

Without Container-to-Container Networking

The diagram below illustrates how two app instances communicate in a deployment without Container-to-Container Networking enabled. Traffic from **App A** must route out and back in through the Gorouter, which restricts performance and the protocol used to send the traffic. In this scenario, **App B** does not know the real source of the traffic it receives and must trust all inbound traffic.

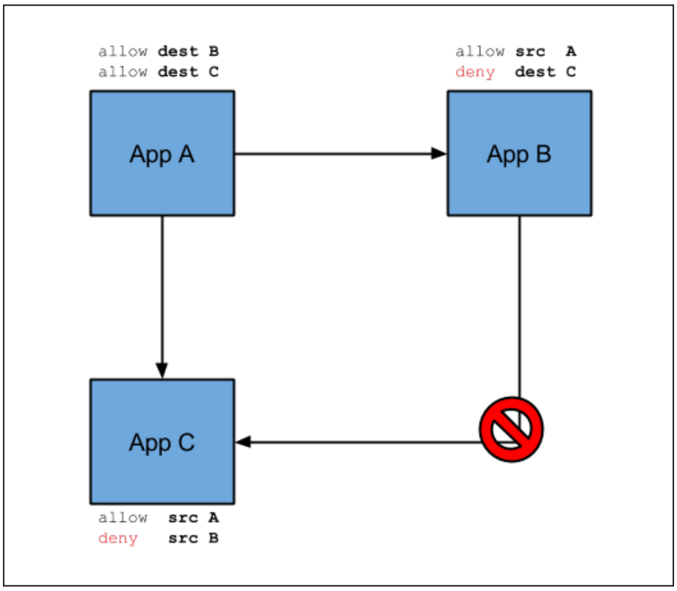


With Container-to-Container Networking

The diagram below illustrates how app instances communicate in a deployment with Container-to-Container Networking enabled. In this example, the operator creates two policies to regulate the flow of traffic between **App A**, **App B**, and **App C**.

- Allow traffic from **App A** to **App B**
- Allow traffic from **App A** to **App C**

If traffic and its direction is not explicitly allowed, it is denied. For example, **App B** cannot send traffic to **App C**.



Container-to-Container Networking versus ASGs

Both application security groups (ASGs) and Container-to-Container Networking policies affect traffic from app instances. The following table highlights differences between ASGs and Container-to-Container Networking policies.

	ASGs	Container-to-Container Networking Policies
Policy granularity	From a space to an IP address range	From a source app to a destination app
Scope	For a space, org, or deployment	For app to app only

Traffic direction	Outbound control	Policies apply for incoming packets from other app instances
Source app	Is not known	Is identified because of direct addressability
Policies take affect	After app restart	Immediately

Policies

Enabling Container-to-Container Networking for your deployment allows you to create policies for communication between app instances. The Container-to-Container Networking feature also provides a unique IP address to each app container and provides direct IP reachability between app instances.

The policies you create specify a source app, destination app, protocol, and port so that app instances can communicate directly without going through the Gorouter, a load balancer, or a firewall. Container-to-Container Networking supports UDP and TCP, and you can configure policies for multiple ports. These policies apply immediately without having to restart the app.

Alternative Network Stacks

The BOSH release that contains the Container-to-Container Networking feature is composed of a pluggable network stack. Advanced users or third-party vendors can integrate a different network stack. For more information about third-party plugins, see the Container-to-Container Networking BOSH release documentation.

Understanding Application Security Groups

Page last updated:

Introduction

This topic provides an overview of Application Security Groups (ASGs), and describes how to manage and administer them. Many of the steps below require the Cloud Foundry Command Line Interface (cf CLI) tool.

Note: If you are creating ASGs for the first time, see Restricting App Access to Internal PCF Components.

Application Security Groups

Application Security Groups (ASGs) are a collections of egress rules that specify the protocols, ports, and IP address ranges where app or task instances send traffic. Because ASGs define **allow** rules, their order of evaluation is unimportant when multiple ASGs apply to the same space or deployment. The platform sets up rules to filter and log outbound network traffic from app and task instances. ASGs apply to both buildpack-based and Docker-based apps and tasks.

Staging vs. Running ASGs

When apps or tasks begin staging, they need traffic rules permissive enough to allow them to pull resources from the network. After an app or task is running, the traffic rules can be more restrictive and secure. To distinguish between these two security requirements, administrators can define a `staging` ASG for app and task staging, and a `running` ASG for app and task runtime.

Platform-wide vs. Space-scoped ASGs

To provide granular control when securing a deployment, an administrator can assign platform-wide ASGs that apply to all app and task instances for the entire deployment, or space-scoped ASGs that apply only to apps and tasks in a particular space.

Simplifying ASGs with a Services Subnet

ASGs can be complicated to configure correctly, especially when the specific IP addresses listed in a group change. To simplify securing a deployment while still letting apps reach external services, operators can deploy the services into a subnet that is separate from their Cloud Foundry deployment. Then the operators can create ASGs for the apps that whitelist those service subnets, while denying access to any virtual machine (VM) hosting other apps.

For examples of typical ASGs, see the Typical Application Security Groups section of this topic.

Default ASGs

Pivotal Application Service (PAS) defines one default ASG, `default\_security\_group`. This group allows all outbound traffic from application containers on public and private networks except for the link-local range, `169.254.0.0/16`, which is blocked.

WARNING: For security, PAS administrators must modify the default ASGs so that outbound network traffic cannot access internal components.

The ASG is defined in the Cloud Controller configuration as follows:

```
security_group_definitions:
- name: default_security_group
  rules:
  - protocol: all
    destination: 0.0.0.0-169.253.255.255
  - protocol: all
    destination: 169.255.0.0-255.255.255.255
```

ASG Sets

ASGs are applied by configuring ASG sets differentiated by *scope*, platform-wide or space specific, and *lifecycle*, staging or running.

Currently, four ASG sets exist in Cloud Foundry:

- Platform-wide staging ASG set, also called “default-staging”
- Platform-wide running ASG set, also called “default-running”
- Space-scoped staging ASG set
- Space-scoped running ASG set

The following table indicates the differences between the four sets.

When an ASG is bound to the…	the ASG rules are applied to…
Platform-wide staging ASG set	the staging lifecycle for all apps and tasks.
Platform-wide running ASG set	the running lifecycle for all app and task instances.
Space-scoped staging ASG set	the staging lifecycle for apps and tasks in a particular space.
Space-scoped running ASG set	the running lifecycle for app and task instances in a particular space.

Typically, ASGs applied during the staging lifecycle are more permissive than the ASGs applied during the running lifecycle. This is because staging often requires access to different resources, such as dependencies.

You use different commands to apply an ASG to each of the four sets. For more information, see the Procedures section of this topic.

Note: To apply a staging ASG to apps within a space, you must use cf CLI 6.28.0 or later.

The Structure and Attributes of ASGs

ASG rules are specified as a JSON array of ASG objects. An ASG object has the following attributes:

Attribute	Description	Notes
`protocol`	`tcp`, `udp`, `icmp`, or `all`	Required
`destination`	A single IP address, an IP address range like `192.0.2.0-192.0.2.50`, or a CIDR block that can receive traffic	
`ports`	A single port, multiple comma-separated ports, or a single range of ports that can receive traffic. Examples: `443`, `80,8080,8081`, `8080-8081`	Required when `protocol` is `tcp` or `udp`
`code`	ICMP code	Required when `protocol` is `icmp`. A value of `-1` allows all codes.
`type`	ICMP type	Required when `protocol` is `icmp`. A value of `-1` allows all types.
`log`	Set to `true` to enable logging. For more information about how to configure system logs to be sent to a syslog drain, see the Using Log Management Services topic.	Logging is only supported with protocol type `tcp`
`description`	An optional text field for operators managing security group rules	

Process for Administering ASGs

The following table outlines the flow of tasks that the administrator carries out over the lifecycle of ASGs. Procedures for each of these tasks are given in Managing ASGs with the cf CLI below.

Note: If you are creating ASGs for the first time, see Restricting App Access to Internal PCF Components.

	Task	For more information, see
1.	Review the existing ASGs. If this is a new deployment, these consist of only the Default ASGs.	View ASGs
2.	Create new ASGs.	Create ASGs

3. 4.	~~Update the existing ASGs.~~ Bind ASGs to an ASG set.	Update ASGs Bind ASGs
5.	If you need to delete an ASG, first unbind it, then delete it.	Unbind ASGs Delete ASGs

Managing ASGs with the cf CLI

This section provides the commands you need to create and manage ASGs.

View ASGs

Run the following cf CLI commands to view information about existing ASGs:

Command	Output
`cf security-groups`	All ASGs
`cf staging-security-groups`	All ASGs applied to the platform-wide staging ASG set
`cf running-security-groups`	All ASGs applied to the platform-wide running ASG set
`cf security-group SECURITY-GROUP`	All rules in the ASG named `SECURITY-GROUP`, for example, `cf security-group dns`

Note: You can also view ASGs in Apps Manager under the **Settings** tab of a space or an app.

Create ASGs

To create an ASG, perform the following steps:

1. Create a rules file: a JSON-formatted single array containing objects that describe the rules. See the following example, which allows ICMP traffic of code `1` and type `0` to all destinations, and TCP traffic to `10.0.11.0/24` on ports `80` and `443`. Also see The Structure and Attributes of ASGs.

```
[
  {
    "protocol": "icmp",
    "destination": "0.0.0.0/0",
    "type": 0,
    "code": 0
  },
  {
    "protocol": "tcp",
    "destination": "10.0.11.0/24",
    "ports": "80,443",
    "log": true,
    "description": "Allow http and https traffic from ZoneA"
  }
]
```

2. Run `cf create-security-group SECURITY-GROUP PATH-TO-RULES-FILE`. Replace `SECURITY-GROUP` with the name of your security group, and `PATH-TO-RULES-FILE` with the absolute or relative path to a rules file.

In the following example, `my-asg` is the name of a security group, and `~/workspace/my-asg.json` is the path to a rules file.

```
$ cf create-security-group my-asg ~/workspace/my-asg.json
```

After the ASG is created, you must bind it to an ASG set before it takes effect. See Bind ASGs below.

Bind ASGs

Note: Binding an ASG does not affect started apps until you restart them. To restart all of the apps in an org or a space, use the app-restarter cf CLI plugin.

To apply an ASG, you must first bind it to an ASG set.

To bind an ASG to the platform-wide staging ASG set, run `cf bind-staging-security-group SECURITY-GROUP`. Replace `SECURITY-GROUP` with the name of your security group.

Example:

```
$ cf bind-staging-security-group my-asg
```

To bind an ASG to the platform-wide running ASG set, run `cf bind-running-security-group SECURITY-GROUP` command. Replace `SECURITY-GROUP` with the name of your security group.

Example:

```
$ cf bind-running-security-group my-asg
```

To bind an ASG to a space-scoped running ASG set, run `cf bind-security-group SECURITY-GROUP ORG SPACE`. Replace `SECURITY-GROUP` with the name of your security group. Replace `ORG` and `SPACE` with the org and space where you want to bind the ASG set.

Example:

```
$ cf bind-security-group my-asg my-org my-space
```

To bind an ASG to a space-scoped staging ASG set, run `cf bind-security-group SECURITY-GROUP ORG SPACE --lifecycle staging`. Replace `SECURITY-GROUP` with the name of your security group. Replace `ORG` and `SPACE` with the org and space where you want to bind the ASG set.

Update ASGs

To update an existing ASG, perform the following steps.

1. Edit the ASG rules in the JSON file.
2. Run `cf update-security-group SECURITY-GROUP PATH-TO-RULES-FILE`. Replace `SECURITY-GROUP` with the name of the existing ASG you want to change, and `PATH-TO-RULES-FILE` with the absolute or relative path to a rules file.

In the following example, `my-asg` is the name of a security group, and `~/workspace/my-asg-v2.json` is the path to a rules file.

```
$ cf update-security-group my-asg ~/workspace/my-asg-v2.json
```

Note: Updating an ASG does not affect started apps until you restart them. To restart all of the apps in an org or a space, use the app-restarter cf CLI plugin.

Unbind ASGs

Note: Unbinding an ASG does not affect started apps until you restart them. To restart all of the apps in an org or a space, use the app-restarter cf CLI plugin.

To unbind an ASG from the platform-wide staging ASG set, run `cf unbind-staging-security-group SECURITY-GROUP`. Replace `SECURITY-GROUP` with the name of your security group.

Example:

```
$ cf unbind-staging-security-group my-asg
```

To unbind an ASG from the platform-wide running ASG set, run `cf unbind-running-security-group SECURITY-GROUP`. Replace `SECURITY-GROUP` with the name of your security group.

Example:

```
$ cf unbind-running-security-group my-asg
```

To unbind an ASG from a specific space, run `cf unbind-security-group SECURITY-GROUP ORG SPACE --lifecycle running`. Replace `SECURITY-GROUP` with the name of your security group. Replace `ORG` and `SPACE` with the org and space where you want to unbind the ASG set, and replace `running` with `staging` if you want to unbind from the staging ASG set.

Example:

```
$ cf unbind-security-group my-asg my-org my-space --lifecycle staging
```

Delete ASGs

Note: You can only delete unbound ASGs. To unbind ASGs, see Unbind ASGs above.

To delete an ASG, run `cf delete-security-group SECURITY-GROUP`. Replace `SECURITY-GROUP` with the name of your security group.

Example:

```
$ cf delete-security-group my-asg
```

Typical ASGs

Below are examples of typical ASGs. Configure your ASGs in accordance with your organization's network access policy for untrusted apps.

ASG	For access to
`dns`	DNS, either public or private
`public-networks`	Public networks, excluding IaaS metadata endpoints
`private-networks`	Private networks in accordance with RFC-1918
`load-balancers`	The internal Pivotal Application Service load balancer and others
`internal-proxies`	Internal proxies
`internal-databases`	Internal databases

DNS

To resolve hostnames to IP addresses, apps require DNS server connectivity, which typically use port 53. Administrators should create or update a `dns` ASG with appropriate rules. Administrators may further restrict the DNS servers to specific IP addresses or ranges of IP addresses.

Example `dns` ASG:

```
[
  {
   "protocol": "tcp",
   "destination": "0.0.0.0/0",
   "ports": "53"
  },
  {
   "protocol": "udp",
   "destination": "0.0.0.0/0",
   "ports": "53"
  }
]
```

Public Networks

Apps often require public network connectivity to retrieve app dependencies, or to integrate with services available on public networks. Example app dependencies include public Maven repositories, NPM, RubyGems, and Docker registries.

Note: You should exclude IaaS metadata endpoints, such as `169.254.169.254`, because the metadata endpoint can expose sensitive environment information to untrusted apps. The `public_networks` example below accounts for this recommendation.

Example `public_networks` ASG:

```
[
  {
   "destination": "0.0.0.0-9.255.255.255",
   "protocol": "all"
  },
  {
   "destination": "11.0.0.0-169.253.255.255",
   "protocol": "all"
  },
  {
   "destination": "169.255.0.0-172.15.255.255",
   "protocol": "all"
  },
  {
   "destination": "172.32.0.0-192.167.255.255",
   "protocol": "all"
  },
  {
   "destination": "192.169.0.0-255.255.255.255",
   "protocol": "all"
  }
]
```

Private Networks

Network connections that are commonly allowable in private networks include endpoints such as proxy servers, Docker registries, load balancers, databases, messaging servers, directory servers, and file servers. Configure appropriate private network ASGs as appropriate. You may find it helpful to use a naming convention with `private_networks` as part of the ASG name, such as `private_networks_databases`.

Note: You should exclude any private networks and IP addresses that app and task instances should not have access to.

Example `private_networks` ASG:

```
[
  {
   "protocol": "tcp",
   "destination": "10.0.0.0-10.255.255.255",
   "ports": "443"
  },
  {
   "protocol": "tcp",
   "destination": "172.16.0.0-172.31.255.255",
   "ports": "443"
  },
  {
   "protocol": "tcp",
   "destination": "192.168.0.0-192.168.255.255",
   "ports": "443"
  }
]
```

Marketplace Services

Each installed Marketplace Service requires its own set of ASG rules to function properly. See the installation instructions for each installed Marketplace Service to determine which ASG rules it requires. For more information about how to provision and integrate services, see the Services Overview topics.

About the ASG Creator Tool

The ASG Creator is a command line tool that you can use to create JSON rules files. The ASG Creator lets you specify IP addresses, CIDRs, and IP address ranges that you want to disallow traffic to, as well as the addresses that you want to allow traffic to. Based on these disallow/allow (exclude/include) lists that you provide as input, the ASG Creator formulates a JSON file of allow rules.

In turn, the JSON file is the input for the `cf create-security-group` command that creates an ASG.

You can download the latest release of the ASG Creator from the Cloud Foundry incubator repository on Github: https://github.com/cloudfoundry-incubator/asg-creator/releases/latest

ASG Logging

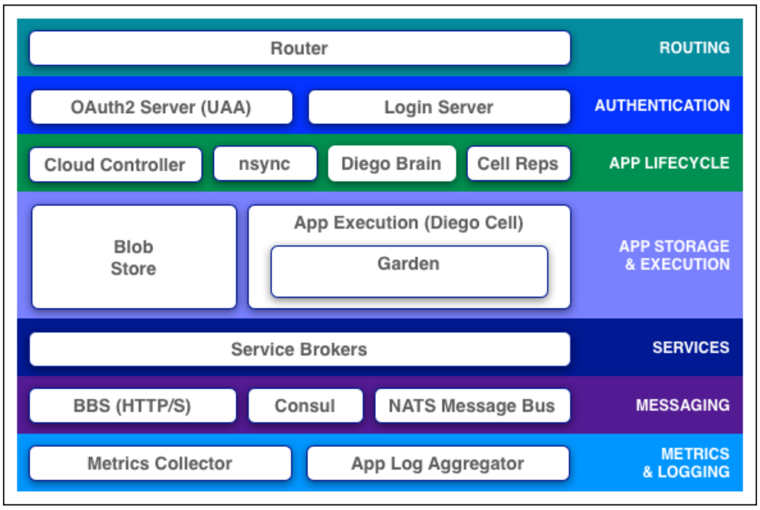
The KB article How to use Application Security Group (ASG) logging describes how you can use ASGs to correlate emitted logs back to an app.

Cloud Foundry Components

Page last updated:

Cloud Foundry components include a self-service application execution engine, an automation engine for application deployment and lifecycle management, and a scriptable command line interface (CLI), as well as integration with development tools to ease deployment processes. Cloud Foundry has an open architecture that includes a buildpack mechanism for adding frameworks, an application services interface, and a cloud provider interface.

Refer to the descriptions below for more information about Cloud Foundry components. Some descriptions include links to more detailed documentation.



Routing

Router

The router routes incoming traffic to the appropriate component, either a Cloud Controller component or a hosted application running on a Diego Cell.

The router periodically queries the Diego Bulletin Board System (BBS) to determine which cells and containers each application currently runs on. Using this information, the router recomputes new routing tables based on the IP addresses of each cell virtual machine (VM) and the host-side port numbers for the cell's containers.

Authentication

OAuth2 Server (UAA) and Login Server

The OAuth2 server (the UAA) and Login Server work together to provide identity management.

App Lifecycle

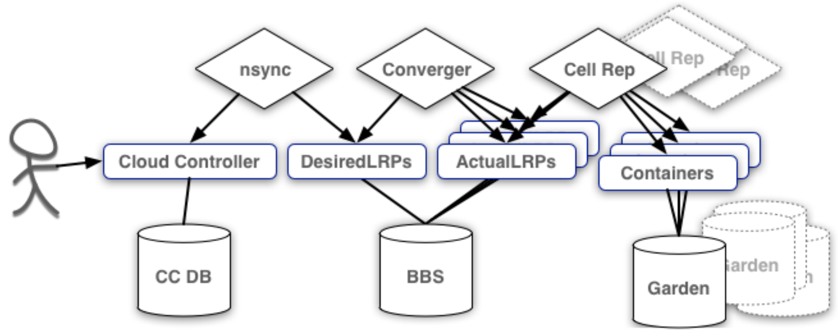
Cloud Controller and Diego Brain

The Cloud Controller (CC) directs the deployment of applications. To push an app to Cloud Foundry, you target the Cloud Controller. The Cloud Controller then directs the Diego Brain through the CC-Bridge components to coordinate individual Diego cells to stage and run applications.

The Cloud Controller also maintain records of orgs, spaces, user roles, services, and more.

nsync, BBS, and Cell Reps

To keep applications available, cloud deployments must constantly monitor their states and reconcile them with their expected states, starting and stopping processes as required.



App Monitoring and Syncing with Diego

The nsync, BBS, and Cell Rep components work together along a chain to keep apps running. At one end is the user. At the other end are the instances of applications running on widely-distributed VMs, which may crash or become unavailable.

Here is how the components work together:

- **nsync** receives a message from the Cloud Controller when the user scales an app. It writes the number of instances into a `DesiredLRP` structure in the Diego BBS database.
- **BBS** uses its convergence process to monitor the `DesiredLRP` and `ActualLRP` values. It launches or kills application instances as appropriate to ensure the `ActualLRP` count matches the `DesiredLRP` count.
- **Cell Rep** monitors the containers and provides the `ActualLRP` value.

App Storage and Execution

Blobstore

The blobstore is a repository for large binary files, which Github cannot easily manage because Github is designed for code. The blobstore contains the following:

- Application code packages
- Buildpacks
- Droplets

You can configure the blobstore as either an internal server or an external S3 or S3-compatible endpoint. See this Knowledge Base article for more information about the blobstore.

Diego Cell

Application instances, application tasks, and staging tasks all run as Garden containers on the Diego Cell VMs. The Diego cell rep component manages the lifecycle of those containers and the processes running in them, reports their status to the Diego BBS, and emits their logs and metrics to Loggregator.

Services

Service Brokers

Applications typically depend on services such as databases or third-party SaaS providers. When a developer provisions and binds a service to an application, the service broker for that service is responsible for providing the service instance.

Messaging

Consul and BBS

Cloud Foundry component VMs communicate with each other internally through HTTP and HTTPS protocols, sharing temporary messages and data stored in two locations:

- A Consul server stores longer-lived control data, such as component IP addresses and distributed locks that prevent components from duplicating actions.
- Diego's Bulletin Board System (BBS) stores more frequently updated and disposable data such as cell and application status, unallocated work, and heartbeat messages. The BBS stores data in MySQL, using the Go MySQL Driver.

The route-emitter component uses the NATS protocol to broadcast the latest routing tables to the routers.

Metrics and Logging

Loggregator

The Loggregator (log aggregator) system streams application logs to developers.

Component: Cloud Controller

Page last updated:

The Cloud Controller provides REST API endpoints for clients to access the system. The Cloud Controller maintains a database with tables for orgs, spaces, services, user roles, and more.

Diego Auction

The Cloud Controller uses the Diego Auction to balance application processes over the cells in a Cloud Foundry installation.

Database (CC_DB)

The Cloud Controller database has been tested with MySQL.

Blobstore

To stage and run apps, Cloud Foundry manages and stores the following types of binary large object (blob) files:

Blob Type	Description	Location in Blobstore
App Packages	Full contents of app directories, including source code and resource files, zipped into single blob files.	`/cc-packages`
Buildpacks	Buildpack directories, which Diego cells download to compile and stage apps with.	`/cc-buildpacks`
Resource Cache	Large files from app packages that the Cloud Controller stores with a SHA for later re-use. To save bandwidth, the Cloud Foundry Command Line Interface (cf CLI) only uploads large application files that the Cloud Controller has not already stored in the resource cache.	`/cc-resources`
Buildpack Cache	Large files that buildpacks generate during staging, stored for later re-use. This cache lets buildpacks run more quickly when staging apps that have been staged previously.	`cc-droplets/buildpack_cache`
Droplets	Staged apps packaged with everything needed to run in a container.	`/cc-droplets`

Cloud Foundry blobstores use the Fog Ruby gem to store blobs in services like Amazon S3, WebDAV, or the NFS filesystem. The file system location of an internal blobstore is `/var/vcap/store/shared`.

A single blobstore typically stores all five types of blobs, but you can configure the Cloud Controller to use separate blobstores for each type.

Automatic Blob Cleanup

After a blob deletion fails silently or something else goes wrong, the blobstore may contain blobs that the Cloud Controller no longer needs or lists in its database. These are called orphan blobs, and they waste blobstore capacity.

The Cloud Controller detects and removes orphan blobs by scanning part of the blobstore daily and checking for any blobs that its database does not account for. The process scans through the entire blobstore every week, and only removes blobs that show as orphans for three consecutive days.

The Cloud Controller performs this automatic cleanup when the `cloud_controller_worker` job property `cc.perform_blob_cleanup` is set to `true`.

Manual Blob Cleanup

The Cloud Controller does not track resource cache and buildpack cache blob types in its database, so it does not clean them up automatically as it does with app package, buildpack, and droplet type blobs.

To clean up the buildpack cache, admin users can run `cf curl -X DELETE /v2/blobstores/buildpack_cache`. This empties the buildpack cache completely, which is a safe operation.

To clean up the resource cache, delete it as follows:

- **Internal blobstore**: Run `bosh ssh` to connect to the blobstore VM (NFS or WebDav) and `rm *` the contents of the `/var/vcap/store/shared/cc-resources` directory.
- **External blobstore**: Use the file store's API to delete the contents of the `resources` bucket.

Do not manually delete app package, buildpack, or droplet blobs from the blobstore. To free up resources from those locations, run `cf delete-buildpack` for buildpacks or `cf delete` for app packages and droplets.

Testing

By default `rspec` runs a test suite with the SQLite in-memory database. Specify a connection string using the `DB_CONNECTION` environment variable to test against MySQL. For example:

```
DB_CONNECTION="mysql2://root:password@localhost:3306/ccng" rspec
```

Component: Messaging (NATS)

Page last updated:

This information was adapted from the NATS README. NATS is a lightweight publish-subscribe and distributed queueing messaging system written in Ruby. Ops Manager sends all NATS traffic using Transport Layer Security (TLS) encryption by default.

Install NATS

```
$ gem install nats
# or
$ rake geminstall

$ nats-sub foo &
$ nats-pub foo 'Hello World!'
```

Basic Usage

```
require "nats/client"

NATS.start do

  # Simple Subscriber
  NATS.subscribe('foo') { |msg| puts "Msg received : '#{msg}'" }

  # Simple Publisher
  NATS.publish('foo.bar.baz', 'Hello World!')

  # Unsubscribing
  sid = NATS.subscribe('bar') { |msg| puts "Msg received : '#{msg}'" }
  NATS.unsubscribe(sid)

  # Requests
  NATS.request('help') { |response| puts "Got a response: '#{response}'" }

  # Replies
  NATS.subscribe('help') { |msg, reply| NATS.publish(reply, "I'll help!") }

  # Stop using NATS.stop, exits EM loop if NATS.start started the loop
  NATS.stop

end
```

Wildcard Subscriptions

```
# "*" matches any token, at any level of the subject.
NATS.subscribe('foo.*.baz') { |msg, reply, sub| puts "Msg received on [#{sub}] : '#{msg}'" }
NATS.subscribe('foo.bar.*') { |msg, reply, sub| puts "Msg received on [#{sub}] : '#{msg}'" }
NATS.subscribe('*.bar.*')   { |msg, reply, sub| puts "Msg received on [#{sub}] : '#{msg}'" }

# ">" matches any length of the tail of a subject and can only be the last token
# E.g. 'foo.>' will match 'foo.bar', 'foo.bar.baz', 'foo.foo.bar.bax.22'
NATS.subscribe('foo.>') { |msg, reply, sub| puts "Msg received on [#{sub}] : '#{msg}'" }
```

Queues Groups

```
# All subscriptions with the same queue name will form a queue group
# Each message will be delivered to only one subscriber per queue group, queuing semantics
# You can have as many queue groups as you want
# Normal subscribers will continue to work as expected.
NATS.subscribe(subject, :queue => 'job.workers') { |msg| puts "Received '#{msg}'" }
```

Advanced Usage

```
# Publish with closure, callback fires when server has processed the message
NATS.publish('foo', 'You done?') { puts 'msg processed!' }

# Timeouts for subscriptions
sid = NATS.subscribe('foo') { received += 1 }
NATS.timeout(sid, TIMEOUT_IN_SECS) { timeout_recvd = true }

# Timeout unless a certain number of messages have been received
NATS.timeout(sid, TIMEOUT_IN_SECS, :expected => 2) { timeout_recvd = true }

# Auto-unsubscribe after MAX_WANTED messages received
NATS.unsubscribe(sid, MAX_WANTED)

# Multiple connections
NATS.subscribe('test') do |msg|
  puts "received msg"
  NATS.stop
end

# Form second connection to send message on
NATS.connect { NATS.publish('test', 'Hello World!') }
```

Component: Gorouter

Page last updated:

The Gorouter routes traffic coming into Cloud Foundry to the appropriate component, whether it is an operator addressing the Cloud Controller or an application user accessing an app running on a Diego Cell. The router is implemented in Go. Implementing a custom router in Go gives the router full control over every connection, which simplifies support for WebSockets and other types of traffic (for example, through HTTP CONNECT). A single process contains all routing logic, removing unnecessary latency.

Refer to the following instructions for help getting started with the Gorouter in a standalone environment.

Setup

```
$ git clone https://github.com/cloudfoundry/gorouter.git
$ cd gorouter
$ git submodule update --init
$ ./bin/go install router/router
$ gem install nats
```

Start

```
# Start NATS server in daemon mode
$ nats-server -d

# Start gorouter
$ ./bin/router
```

Usage

The Gorouter receives route updates though NATS. By default, routes that have not been updated in two minutes are pruned. Therefore, to maintain an active route, you must ensure that the route is updated at least every two minutes. The format of these route updates is as follows:

```
{
  "host": "127.0.0.1",
  "port": 4567,
  "uris": [
    "my_first_url.vcap.me",
    "my_second_url.vcap.me"
  ],
  "tags": {
    "another_key": "another_value",
    "some_key": "some_value"
  }
}
```

Such a message can be sent to both the `router.register` subject to register URIs, and to the `router.unregister` subject to unregister URIs, respectively.

```
$ nohup ruby -rsinatra -e 'get("/") { "Hello!" }' &
$ nats-pub 'router.register' '{"host":"127.0.0.1","port":4567,
    "uris":["my_first_url.vcap.me","my_second_url.vcap.me"],
    "tags":{"another_key":"another_value","some_key":"some_value"}}'
Published [router.register] : '{"host":"127.0.0.1","port":4567,
    "uris":["my_first_url.vcap.me","my_second_url.vcap.me"],
    "tags":{"another_key":"another_value","some_key":"some_value"}}'
$ curl my_first_url.vcap.me:8080
Hello!
```

Instrumentation

The Gorouter provides `/varz` and `/healthz` http endpoints for monitoring.

The `/routes` endpoint returns the entire routing table as JSON. Each route has an associated array of `host:port` entries.

All of the endpoints require http basic authentication, credentials for which you can acquire through NATS. You can explicitly set the `port`, `user` and password (`pass` is the config attribute) in the gorouter.yml config file `status` section.

```
status:
  port: 8080
  user: some_user
  pass: some_password
```

Example interaction with `curl`:

```
$ curl -vvv "http://someuser:somepass@127.0.0.1:8080/routes"
* About to connect() to 127.0.0.1 port 8080 (#0)
*   Trying 127.0.0.1...
* Connected
* Connected to 127.0.0.1 (127.0.0.1) port 8080 (#0)
* Server auth using Basic with user 'someuser'
> GET /routes HTTP/1.1
> Authorization: Basic c29tZXVzZXI6c29tZXBhc3M=
> User-Agent: curl/7.24.0 (x86_64-apple-darwin12.0) libcurl/7.24.0 OpenSSL/0.9.8r zlib/1.2.5
> Host: 127.0.0.1:8080
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Date: Mon, 25 Mar 2013 20:31:27 GMT
< Transfer-Encoding: chunked
<
{"0295dd314aaf582f201e655cbd74ade5.cloudfoundry.me":["127.0.0.1:34567"],
"03e316d6aa375d1dc1153700da5f1798.cloudfoundry.me":["127.0.0.1:34568"]}
```

Pivotal

Component: User Account and Authentication (UAA) Server

Page last updated:

This topic provides an overview of the User Account and Authentication (UAA) Server, the identity management service for Cloud Foundry

The primary role of the UAA is as an OAuth2 provider, issuing tokens for client apps to use when they act on behalf of Cloud Foundry users. In collaboration with the login server, the UAA can authenticate users with their Cloud Foundry credentials, and can act as an SSO service using those, or other, credentials.

The UAA has endpoints for managing user accounts and for registering OAuth2 clients, as well as various other management functions.

Quick Start

You can deploy the UAA locally or to Cloud Foundry.

- Deploy UAA Locally
- Deploy UAA to Cloud Foundry

Deploy UAA Locally

Follow the instructions below to deploy and run the UAA locally.

1. In a terminal window, clone the UAA GitHub repository.

```
$ git clone git://github.com/cloudfoundry/uaa.git
```

2. Navigate to the directory where you cloned the UAA GitHub repository, then run the `./gradlew run` command to build and run all the components that comprise the UAA and the example programs, `uaa`, `samples/api`, and `samples/app`.

```
$ cd uaa
$ ./gradlew run
```

3. If successful, the three apps run together on a single instance of Tomcat listening on port 8080, with endpoints `/uaa`, `/app`, and `/api`.

Use Local UAA

Follow the steps below to access and use a locally-deployed UAA server.

1. Run the UAA server as described in the Deploy Locally section, above.

2. Open another terminal window. From the project base directory, run `curl -H "Accept: application/json" localhost:8080/uaa/login` to query the login endpoint about the system. For example:

```
$ curl -H "Accept: application/json" localhost:8080/uaa/login
{
  "timestamp":"2012-03-28T18:25:49+0100",
  "app" : {"version":"1.8.3"},
  "commit_id":"cba0958",
  "prompts":{"username":["text","Email"],
    "password":["password","Password"]
  }
}
```

3. Run `gem install cf-uaac` to install the Cloud Foundry UAA Command Line Client (UAAC) Ruby gem.

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

4. Run `uaac target http://localhost:8080/uaa` to target the local UAA Server endpoint.


```
$ uaac target http://localhost:8080/uaa
```

5. Run `uaac token get USERNAME PASSWORD` to log in. Replace `USERNAME` with your user name, and `PASSWORD` with your password. For example:

```
$ uaac token get marissa koala
```

If you do not specify a username and password, the UAAC prompts you to supply them.

The `uaac token client get` command authenticates and obtains an access token from the server using the OAuth2 implicit grant, similar to the approach intended for a standalone client like the Cloud Foundry Command Line Interface (cf CLI).

UAAC stores the access token in the `~/.uaac.yml` file. Open the `~/.uaac.yml` in a text editor and copy this access token to use in the next step.

6. Run `uaac token decode ACCESS-TOKEN-VALUE` to retrieve the token details. Replace `ACCESS-TOKEN-VALUE` with your access token, copied from your `~/.uaac.yml` file. The UAAC should display your username and the client id of the original token grant. For example:

```
$ uaac token decode abcdef0123456789ABCDEF0123456789ghijklmnopqr01234567890
jti: e3a7d065-8514-43c2-b1f8-f8ab761791e2
sub: f796d1a2-eca3-4079-a317-f3224f8f7832
scope: scim.userids password.write openid cloud_controller.write cloud_controller.read
client_id: cf
cid: cf
user_id: f796d1a2-eca3-4079-a317-f3224f8f7832
user_name: marissa
email: marissa@example.org
iat: 1413495264
exp: 1413538464
iss: http://localhost:8080/uaa/oauth/token
aud: scim openid cloud_controller password
```

Deploy UAA to Cloud Foundry

Follow the instructions below to build the UAA as an app and push it to Cloud Foundry using the Cloud Foundry Command Line Interface (cf CLI).

1. In a terminal window, clone the UAA GitHub repository.

```
$ git clone git://github.com/cloudfoundry/uaa.git
```

2. Navigate to the directory where you cloned the UAA GitHub repository, then run the `./gradlew :cloudfoundry-identity-uaa:war` command build the UAA as a WAR file.

```
$ cd uaa
$ ./gradlew :cloudfoundry-identity-uaa:war
```

3. Run the cf CLI `cf push APP-NAME -m 512M -p PATH-TO-WAR-FILE --no-start` command to push the app to Cloud Foundry. Replace `APP-NAME` with a name for your UAA app, and `PATH-TO-WAR-FILE` with the path to the WAR file you created in the previous step. For example:

```
$ cf push MYUAA -m 512M -p uaa/build/libs/cloudfoundry-identity-uaa-1.8.0.war --no-start
```

4. Run `cf set-env APP-NAME SPRING_PROFILES_ACTIVE default` to set the `SPRING_PROFILES_ACTIVE` environment variable with the value `default`. Replace `APP-NAME` with the name of your app that you used in the previous step. For example:

```
$ cf set-env MYUAA SPRING_PROFILES_ACTIVE default
```

5. Run `cf start APP-NAME` to start your app. Replace `APP-NAME` with the name of your app. For example:

```
$ cf start MYUAA
```

Use Remote UAA

You use a UAA server that you pushed as an app to Cloud Foundry in a similar way to one you run locally. You do not need app token encoding because you do not have the client secret.

Pivotal

Follow the steps below to access and use a UAA server that you pushed as an app to Cloud Foundry.

1. Deploy UAA to Cloud Foundry as described above.

2. From the project base directory, run `curl -H "Accept: application/json" APP-FQDN/login` to query the external login endpoint about the system. Replace `APP-FQDN` with the FQDN of your app. For example:

```
$ curl -H "Accept: application/json" uaa.example.org/login
{
 "timestamp":"2014-09-15T18:25:04+0000",
 "app":{"version":"1.8.3"},
 "commit_id":"git-metadata-not-found",
 "prompts":{"username":["text","Email"],
    "password":["password","Password"]
 }
}
```

3. Run `gem install cf-uaac` to install the Cloud Foundry UAA Command Line Client (UAAC) Ruby gem.

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

4. Run `uaac target APP-FQDN` to target the remote UAA Server endpoint. Replace `APP-FQDN` with the FQDN of your app.

```
$ uaac target uaa.example.org
```

5. Run `uaac token get USERNAME PASSWORD` to log in. Replace `USERNAME` with your user name, and `PASSWORD` with your password. For example:

```
$ uaac token get marissa koala
```

If you do not specify a username and password, the UAAC prompts you to supply them.

The `uaac token client get` command authenticates and obtains an access token from the server using the OAuth2 implicit grant, similar to the approach intended for a standalone client like the Cloud Foundry Command Line Interface (cf CLI).

Integration Tests

Run the integration tests with the following command:

```
$ ./gradlew integrationTest
```

This command starts a UAA server running in a local Apache Tomcat instance. By default, the service URL is set to `http://localhost:8080/uaa`.

You can set the environment variable `CLOUD_FOUNDRY_CONFIG_PATH` to a directory containing a `uaa.yml` file where you change the URLs used in the tests, and where you can set the UAA server context root.

Custom YAML Configuration

Follow the steps below to modify the runtime parameters.

1. Create a `uaa.yml` file in the following format:

```
uaa:
 host: UAA-HOSTNAME
 test:
  username: USERNAME
  password: PASSWORD
  email: EMAIL-ADDRESS
```

Replace the values in the above format as follows:
- **UAA-HOSTNAME**: The FQDN of UAA app. Example: `uaa.example.org`
- **USERNAME**: A valid username. Example: `dev@example.org`
- **PASSWORD**: Password for the above username.
- **EMAIL-ADDRESS**: Email address for the above user. Example: `dev@example.org`

2. From the `uaa/uaa` directory, run `CLOUD_FOUNDRY_CONFIG_PATH=/tmp ./gradlew test`.

3. The web app looks for a YAML file in the following locations when it starts, with later entries overriding earlier ones:

```
classpath:uaa.yml
file:${CLOUD_FOUNDRY_CONFIG_PATH}/uaa.yml
file:${UAA_CONFIG_FILE}
${UAA_CONFIG_URL}
```

Test with PostgreSQL or MySQL

The default UAA unit tests, `./gradlew test`, use HyperSQL (hsqldb).

To use a different database management system, create a `uaa.yml` file containing `spring_profiles: default,OTHER-DBMS` in the `src/main/resources/` directory. Replace `OTHER-DBMS` with the name of the other database management system to use.

For example, run the following command to run the unit tests using PostgreSQL instead of hsqldb:

```
$ echo "spring_profiles: default,postgresql" > src/main/resources/uaa.yml
$ ./gradlew test integrationTest
```

Run the following command to run the unit tests using MySQL instead of hsqldb:

```
$ echo "spring_profiles: default,mysql" > src/main/resources/uaa.yml
$ ./gradlew test integrationTest
```

You can find the database configuration for the common and scim modules at `common/src/test/resources/(mysql|postgresql).properties` and `scim/src/test/resources/(mysql|postgresql).properties`.

UAA Projects

The following UAA projects exist:

- `common`: A module containing a JAR with all the business logic. `common` is used in the web apps listed below.
- `uaa`: The UAA server. `uaa` provides an authentication service and authorized delegation for back-end services and apps by issuing OAuth2 access tokens.
- `api`: A sample OAuth2 resource service that returns a mock list of deployed apps. `api` provides resources that other apps might want to access on behalf of the resource owner.
- `app`: A sample user app that uses both `api` and `uaa`. `app` is a web app that requires single sign-on and access to the `api` service on behalf of users.
- `scim`: The SCIM user management module used by UAA.

UAA Server

The authentication service, `uaa`, is a Spring MVC web app. You can deploy it in Tomcat or your container of choice, or execute `./gradlew run` to run it directly from the `uaa` directory in the source tree. When run with Gradle, `uaa` listens on port `8080` and has URL `http://localhost:8080/uaa`.

The UAA Server supports the APIs defined in the UAA-APIs document which include the following:

- The OAuth2 `/authorize` and `/token` endpoints
- A `/login_info` endpoint to allow querying for required login prompts
- A `/check_token` endpoint to allow resource servers to obtain information about an access token submitted by an OAuth2 client
- SCIM user provisioning endpoint
- OpenID connect endpoints to support authentication `/userinfo` and `/check_id`

Command line clients can perform authentication by submitting credentials directly to the `/authorize` endpoint.

An `ImplicitAccessTokenProvider` exists in Spring Security OAuth to use if your client is Java.

By default, `uaa` will launch with a context root `/uaa`.

Configuration

A `uaa.yml` file exists in the app. This file provides defaults to the placeholders in the Spring XML.

You can override any occurrences of `${placeholder.name}` in the XML by adding it to the `uaa.yml` file, or by providing a System property, `-D` to JVM, with the same name.

All passwords and client secrets in the config files are in plain text, but are inserted into the UAA database encrypted with BCrypt.

User Account Data

The default uses an in-memory RDBMS user store, pre-populated with a single test user `marissa` with password `koala`.

To use PostgreSQL for user data, activate the Spring profile `postgresql`.

You can configure the active profiles in `uaa.yml` using the following:

```
spring_profiles: postgresql,default
```

To use PostgreSQL instead of HyperSQL (hsqldb):

```
$ echo "spring_profiles: postgresql,default" > src/main/resources/uaa.yml
$ ./gradlew run
```

To bootstrap a microcloud-type environment, you need an admin client. A database initializer component that inserts an admin client exists. If the default profile is active, for example not `postgresql`, a cf CLI client exists so that the Gem login works with no additional configuration required. You can override the default settings and add additional clients in the `uaa.yml` file:

```
oauth:
  clients:
    admin:
      authorized-grant-types: client_credentials
      scope: read,write,password
      authorities: ROLE_CLIENT,ROLE_ADIN
      id: admin
      secret: adminclientsecret
      resource-ids: clients
```

You can use the admin client to create additional clients. You must have a client with read/write access to the `scim` resource to create user accounts. The integration tests handle this automatically by inserting client and user accounts as necessary for the tests.

Sample Apps

Two sample apps are included with the UAA: `/api` and `/app`.

Run them with `./gradlew run` from the `uaa` root directory. All three apps, `/uaa`, `/api`, and `/app`, are simultaneously deployed.

API Sample App

The `api` sample app is an example resource server. It hosts a service that returns a list of mock apps under `/apps`.

APP Sample App

The `app` sample app is a user interface app, primarily aimed at browsers, that uses OpenID Connect for authentication and OAuth2 for access grants. `app` authenticates with the Auth service, then accesses resources in the API service. Run it with `./gradlew run` from the `uaa` root directory.

The app can operate in multiple different profiles according to the location and presence of the UAA server and the login app. By default, the app looks for a UAA on `localhost:8080/uaa`, but you can change this by setting the `UAA_PROFILE` environment variable or System Property.

The app source code, `samples/app/src/main/resources`, contains multiple properties files pre-configured with different likely locations for those servers. The names of these properties files follow the form `app-UAA_PROFILE.properties`.

The naming convention for the `UAA_PROFILE` is as follows:

- `local`: a localhost deployment
- `vcap`: a `vcap.me` deployment
- `staging`: a staging deployment

Profile names can be hyphenated to indicate multiple contexts. For example, `local-vcap` can be used when the login server is in a different location than the UAA server.

Use Cases

1. See all apps:

   ```
   GET /app/apps
   ```

   Browser is redirected through a series of authentication and access grant steps, and then the photos are shown.

2. See the currently logged in user details, a selection of attributes from the OpenID provider:

   ```
   GET /app
   ```

LOGIN App

The `login` app is a user interface for authentication. The UAA can also authenticate user accounts, but only if it manages them itself, and it only provides a basic UI. You can brand and customize the login app for non-native authentication and for more complicated UI flows, like user registration and password reset.

The login app is itself an OAuth2 endpoint provider, but delegates those features to the UAA server. Therefore, configuration for the login app consists of locating the UAA through its OAuth2 endpoint URLs and registering the login app itself as a client of the UAA. A `login.yml` for the UAA locations exists, such as for a local `vcap` instance:

```
uaa:
  url: http://uaa.vcap.example.net
  token:
    url: http://uaa.vcap.example.net/oauth/token
  login:
    url: http://uaa.vcap.example.net/login.do
```

An environment variable or Java System property also exists, `LOGIN_SECRET`, for the client secret that the app uses when it authenticates itself with the UAA. The login app is registered by default in the UAA only if there are no active Spring profiles. In the UAA, the registration is located in the `oauth-clients.xml` config file:

```
id: login
secret: loginsecret
authorized-grant-types: client_credentials
authorities: ROLE_LOGIN
resource-ids: oauth
```

Use Cases

1. Authenticate:

   ```
   GET /login
   ```

   The sample app presents a form login interface for the backend UAA, and an OpenID widget where a user can authenticate using Google or other credentials.

2. Approve OAuth2 token grant:

```
GET /oauth/authorize?client_id=app&response_type=code...
```

Standard OAuth2 Authorization Endpoint. The UAA handles client credentials and all other features in the back end, and the login app is used to render the UI.

3. Obtain access token:

```
POST /oauth/token
```

Standard OAuth2 Authorization Endpoint passed through to the UAA.

Scopes

UAA covers multiple scopes of privilege, including access to UAA, access to Cloud Controller, and access to the router.

See the tables below for a description of the scopes covered by UAA:

- UAA Scopes
- Cloud Controller Scopes
- Router Scopes
- Other Scopes

UAA Scopes

Scope	Description
`uaa.user`	This scope indicates that this is a user. It is required in the token if submitting a GET request to the OAuth 2 `/authorize` endpoint.
`uaa.none`	This scope indicates that this client will not be performing actions on behalf of a user.
`uaa.admin`	This scopes indicates that this is the superuser.
`scim.write`	This scope gives admin write access to all SCIM endpoints, `/Users`, and `/Groups`.
`scim.read`	This scope gives admin read access to all SCIM endpoints, `/Users`, and `/Groups`.
`scim.create`	This scope gives the ability to create a user with a POST request to the `/Users` endpoint, but not to modify, read, or delete users.
`scim.userids`	This scope is required to convert a username and origin into a user ID and vice versa.
`scim.invite`	This scope is required to participate in invitations using the `/invite_users` endpoint.
`groups.update`	This scope gives the ability to update a group. This ability can also be provided by the broader `scim.write` scope.
`password.write`	This admin scope gives the ability to change a user’s password.
`openid`	This scope is required to access the `/userinfo` endpoint. It is intended for OpenID clients.
`idps.read`	This scope gives read access to retrieve identity providers from the `/identity-providers` endpoint.
`idps.write`	This scope gives the ability to create and update identity providers from the `/identity-providers` endpoint.
`clients.admin`	This scope gives the ability to create, modify, and delete clients.
`clients.write`	This scope is required to create and modify clients. The scopes are prefixed with the scope holder’s client ID. For example, `id:testclient authorities:client.write` gives the ability to create a client that has scopes with the `testclient.` prefix. Authorities are limited to `uaa.resource`.
`clients.read`	This scope gives the ability to read information about clients.
`clients.secret`	This admin scope is required to change the password of a client.
`zones.read`	This scope is required to invoke the `/identity-zones` endpoint to read identity zones.
`zones.write`	This scope is required to invoke the `/identity-zones` endpoint to create and update identity zones.

`scim.zones`	This is a limited scope that only allows adding a user to, or removing a user from, zone management groups under the path `/Groups/zones`.
`oauth.approval`	`/approvals endpoint`. This scope is required to approve or reject clients to act on a user's behalf. This is a default scope defined in the `uaa.yml` file.
`oauth.login`	This scope is used to indicate a login app, such as external login servers, can perform trusted operations, such as creating users not authenticated in the UAA.
`approvals.me`	This scope is not currently used.
`uaa.resource`	This scope indicates that this is a resource server, used for the `/check_token` endpoint.
`zones.ZONE-ID.admin`	This scope permits operations in a designated zone, such as creating identity providers or clients in another zone, by authenticating against the default zone. This scope is used with the `X-Identity-Zone-Id header`.
`zones.ZONE-ID.read`	This scope permits reading the given identity zone. This scope is used with the `X-Identity-Zone-Id header`.
`zones.ZONE-ID.clients.admin`	This scope translates into `clients.admin` after zone switch completes. This scope is used with the `X-Identity-Zone-Id header`.
`zones.ZONE-ID.clients.read`	This scope translates into `clients.read` after zone switch completes. This scope is used with the `X-Identity-Zone-Id header`.
`zones.ZONE-ID.clients.write`	This scope translates into `clients.write` after zone switch completes. This scope is used with the `X-Identity-Zone-Id header`.
`zones.ZONE-ID.clients.scim.read`	This scope translates into `scim.read` after zone switch completes. This scope is used with the `X-Identity-Zone-Id header`.
`zones.ZONE-ID.clients.scim.create`	This scope translates into `scim.create` after zone switch completes. This scope is used with the `X-Identity-Zone-Id header`.
`zones.ZONE-ID.clients.scim.write`	This scope translates into `scim.write` after zone switch completes. This scope is used with the `X-Identity-Zone-Id header`.
`zones.ZONE-ID.idps.read`	This scope translates into `idps.read` after zone switch completes. This scope is used with the `X-Identity-Zone-Id header`.

Cloud Controller Scopes

Scope	Description
`cloud_controller.read`	This scope gives the ability to read from any Cloud Controller route the token has access to.
`cloud_controller.write`	This scope gives the ability to post to Cloud Controller routes the token has access to.
`cloud_controller.admin`	This admin scope gives full permissions to Cloud Controller.
`cloud_controller.admin_read_only`	This admin scope gives read permissions to Cloud Controller.
`cloud_controller.global_auditor`	This scope gives read-only access to all Cloud Controller API resources except for secrets such as environment variables.

Routing Scopes

Scope	Description
`routing.routes.read`	This scope gives the ability to read the full routing table from the router.
`routing.routes.write`	This scope gives the ability to write the full routing table from the router.
`routing.router_groups.read`	This scope gives the ability to read the full list of routing groups.
`routing.router_groups.write`	This scopes gives the ability to write the full list of routing groups.

Other Scopes

Scope	Description
`doppler.firehose`	This scope gives the ability to read logs from the Loggregator Firehose endpoint.

`notifications.write`	This scope gives the ability to send notifications through the Notification Service.

Component: Garden

Page last updated:

This topic describes Garden, the component that Cloud Foundry uses to create and manage isolated environments called containers. Each instance of an application deployed to Cloud Foundry runs within a container. For more information about how containers work, see the Container Mechanics section of the *Understanding Container Security* topic.

Backends

Garden has pluggable backends for different platforms and runtimes, and specifies a set of interfaces that each platform-specific backend must implement. These interfaces contain methods to perform the following actions:

- Create and delete containers
- Apply resource limits to containers
- Open and attach network ports to containers
- Copy files into and out of containers
- Run processes within containers
- Stream `STDOUT` and `STDERR` data out of containers
- Annotate containers with arbitrary metadata
- Snapshot containers for redeploys without downtime

For more information, see the Garden repository on GitHub.

Garden-runC

Cloud Foundry currently uses the Garden-runC backend, a Linux-specific implementation of the Garden interface using the Open Container Interface (OCI) standard. Previous versions of Cloud Foundry used the Garden-Linux backend.

> **Note**: PAS versions v1.8.8 and above use Garden-runC instead of Garden-Linux.

Garden-runC has the following features:

- Uses the same OCI low-level container execution code as Docker and Kubernetes, so container images run identically across all three platforms
- AppArmor is configured and enforced by default for all unprivileged containers
- Seccomp whitelisting restricts the set of system calls a container can access, reducing the risk of container breakout
- Allows pluggable networking and rootfs management

For more information, see the Garden-runC repository on GitHub.

Garden RootFS (GrootFS)

Garden manages container filesystems through a plugin interface. Cloud Foundry uses the GrootFS plugin for this task. GrootFS is a Linux-specific implementation of the Garden volume plugin interface.

GrootFS performs the following actions:

- Creates container filesystems based on buildpacks and droplets
- Creates container filesystems based on remote docker images
- Authenticates with remote registries when using remote images
- Properly maps UID/GID for all files inside an image
- Executes garbage collection to remove unused volumes
- Applies per container disk quotas
- Provides per container disk usage stats

For more information, see the GrootFS repository on GitHub.

HTTP Routing

Page last updated:

This topic describes features of HTTP routing handled by the Gorouter, which is part of the Cloud Foundry (CF) routing tier.

Session Affinity

The Gorouter supports session affinity, or*sticky sessions*, for incoming HTTP requests to compatible apps.

With sticky sessions, when multiple instances of an app are running on CF, requests from a particular client always reach the same app instance. This allows apps to store session data specific to a user session.

- To support sticky sessions, configure your app to return a `JSESSIONID` cookie in responses. The app generates a `JSESSIONID` as a long hash in the following format:

```
1A530637289A03B07199A44E8D531427
```

- If an app returns a `JSESSIONID` cookie to a client request, the CF routing tier generates a unique `VCAP_ID` for the app instance based on its GUID in the following format:

```
323f211e-fea3-4161-9bd1-615392327913
```

- On subsequent requests, the client must provide both the `JSESSIONID` and `VCAP_ID` cookies.

The CF routing tier uses the `VCAP_ID` cookie to forward client requests to the same app instance every time. The `JSESSIONID` cookie is forwarded to the app instance to enable session continuity. If the app instance identified by the `VCAP_ID` crashes, the Gorouter attempts to route the request to a different instance of the app. If the Gorouter finds a healthy instance of the app, it initiates a new sticky session.

Note: CF does not persist or replicate HTTP session data across app instances. If an app instance crashes or is stopped, session data for that instance is lost. If you require session data to persist across crashed or stopped instances, or to be shared by all instances of an app, store session data in a CF marketplace service that offers data persistence.

HTTP Headers

HTTP traffic passed from the Gorouter to an app includes the following HTTP headers:

- `X-Forwarded-Proto` gives the scheme of the HTTP request from the client. The scheme is HTTP if the client made an insecure request or HTTPS if the client made a secure request. Developers can configure their apps to reject insecure requests by inspecting the HTTP headers of incoming traffic and rejecting traffic that includes `X-Forwarded-Proto` with the scheme of HTTP.
- `X-Forwarded-For` gives the IP address of the client originating the request.

If your load balancer terminates TLS upstream from the Gorouter, it must append these headers to requests forwarded to the Gorouter. For more information, see the Securing Traffic into Cloud Foundry topic.

Zipkin Tracing in HTTP Headers

Zipkin is a tracing system that enables app developers to troubleshoot failures or latency issues. Zipkin provides the ability to trace requests and responses across distributed systems. See Zipkin.io for more information.

When the Zipkin feature is enabled in Cloud Foundry, the Gorouter examines the HTTP request headers and performs the following:

- If the `X-B3-TraceId` and `X-B3-SpanId` HTTP headers are not present in the request, the Gorouter generates values for these and inserts the headers into the request forwarded to an application. These values are also found in the Gorouter access log message for the request: `x_b3_traceid` and `x_b3_spanid`.
- If both `X-B3-TraceId` and `X-B3-SpanId` HTTP headers are present in the request, the Gorouter forwards the same value for `X-B3-TraceId`, generates a new value for `X-B3-SpanId`, and adds the `X-B3-ParentSpan` header, and sets to the value of the span id in the request. In addition to these trace and span ids, the Gorouter access log message for the request includes `x_b3_parentspanid`.

Developers can then add Zipkin trace IDs to their application logging in order to trace app requests and responses in Cloud Foundry.

After adding Zipkin HTTP headers to app logs, developers can use `cf logs myapp` to correlate the trace and span ids logged by the Gorouter with the trace ids logged by their app. To correlate trace IDs for a request through multiple apps, each app must forward appropriate values for the headers with requests to other applications.

App Instance Routing in HTTP Headers

Developers who want to obtain debug data for a specific instance of an app can use the HTTP header `X-CF-APP-INSTANCE` to make a request to an app instance.

Perform the following steps to make an HTTP request to a specific app instance:

1. Obtain the GUID of your app:

   ```
   $ cf app YOUR-APP --guid
   ```

2. List your app instances and retrieve the index number of the instance you want to debug:

   ```
   $ cf app YOUR-APP
   ```

3. Make a request to the app route using the HTTP header `X-CF-APP-INSTANCE` set to the concatenated values of the app GUID and the instance index:

   ```
   $ curl app.example.com -H "X-CF-APP-INSTANCE":"YOUR-APP-GUID:YOUR-INSTANCE-INDEX"
   ```

Forward Client Certificate to Applications

Applications that require mutual TLS (mTLS) need metadata from client certificates to authorize requests. Cloud Foundry supports this use case without bypassing layer-7 load balancers and the Gorouter.

The HTTP header `X-Forwarded-Client-Cert` (XFCC) may be used to pass the originating client certificate along the data path to the application. Each component in the data path must trust that the downstream component has not allowed the header to be tampered with.

If you configure the load balancer to terminate TLS and set the XFCC header from the received client certificate, then you must also configure the load balancer to strip this header if it is present in client requests. This configuration is required to prevent spoofing of the client certificate.

The following sections describe supported deployment configurations.

Terminating TLS at Load Balancer

By default, Gorouter forwards arbitrary headers that are not otherwise mentioned in the docs; this includes the XFCC header.

For applications to receive the XFCC header, configure your load balancer to set the XFCC header with the contents of the client certificate received in the TLS handshake.

This mode is enabled when the **TLS terminated for the first time at infrastructure load balancer** option is selected in the **Networking** configuration screen of the PAS tile.

Terminating TLS at HAProxy

This option allows you to configure support for the XFCC header while leveraging HAProxy. When selected, HAProxy sets the XFCC header to the contents of the client certificate received in the TLS handshake.

Selecting this configuration requires that the load balancer in front of HAProxy is configured to pass through the TLS handshake to HAProxy via TCP.

This mode is enabled when the **TLS terminated for the first time at HAProxy** option is selected in the **Networking** configuration screen of the PAS tile.

HAProxy trusts the Diego intermediate certificate authority. This trust is enabled automatically and permits mutual authentication between applications that are running on Pivotal Cloud Foundry.

Terminating TLS at Gorouter

If the Gorouter is the first component to terminate TLS, such that it receives the certificate of the originating client in the mutual TLS handshake, the operator should select this option. When selected, Gorouter sets the XFCC header to the contents of the client certificate received in the TLS handshake and strips the XFCC header when present in a request.

Selecting this configuration requires that the load balancer in front of Gorouter is configured to pass through TLS handshake to Gorouter via TCP.

This mode is enabled when the **TLS terminated for the first time at the Router** option is selected in the **Networking** configuration screen of the PAS tile.

Gorouter trusts the Diego intermediate certificate authority. This trust is enabled automatically and permits mutual authentication between applications that are running on Pivotal Cloud Foundry.

SSL/TLS Termination

Depending on your needs, you can configure your deployment to terminate SSL/TLS at the Gorouter, at the Gorouter and the load balancer, or at the load balancer only. For more information, see the Securing Traffic into Cloud Foundry topic.

Transparent Retries

If the Gorouter cannot establish a TCP connection with a selected application instance, the Gorouter considers the instance ineligible for requests for 30 seconds, and the Gorouter transparently attempts to connect to another application instance. Once the Gorouter has established a TCP connection with an application instance, the Gorouter forwards the HTTP request.

See the Round-Robin Load Balancing section below for more information about how the Gorouter forwards requests to application instances.

Round-Robin Load Balancing

The Gorouter uses the round-robin algorithm for load balancing incoming requests to application instances. The Gorouter maintains a dynamically updated list of application instances for each route, and forwards each request for a given route to the next application instance in the list.

WebSockets

WebSockets is a protocol providing bi-directional communication over a single, long-lived TCP connection, commonly implemented by web clients and servers. WebSockets are initiated through HTTP as an upgrade request. The Gorouter supports this upgrade handshake, and will hold the TCP connection open with the selected application instance. To support WebSockets, the operator must configure the load balancer correctly. Depending on the configuration, clients may have to use a different port for WebSocket connections, such as port 4443, or a different domain name. For more information, see the Supporting WebSockets topic.

Keepalive Connections

From Frontend Clients

The Gorouter supports keepalive connections from clients; it will not close the TCP connection with clients immediately after returning an HTTP response. It is up to clients to close these connections.

To Backend Servers

By default, the Gorouter closes the TCP connection with an app instance or system component after receiving an HTTP response.

When Keepalive Connections is enabled Gorouter will maintain established TCP connections to backends, with a configurable maximum that limits the number of total idle connections from each Gorouter. The Gorouter will reuse idle connections for routing of subsequent requests if one exists for the selected backend, lowering latency and increasing throughput. If no idle connection exists, a new connection will be established, and if the Gorouter limit has not been reached that connection will be maintained. The number of idle connections from each Gorouter to an individual backend is limited to 100.

Operators should see Router Idle Keepalive Connections for details.

How the Diego Auction Allocates Jobs

Page last updated:

This topic provides an overview of the structure and components of Diego, the container management system for Pivotal Cloud Foundry versions 1.6 and newer.

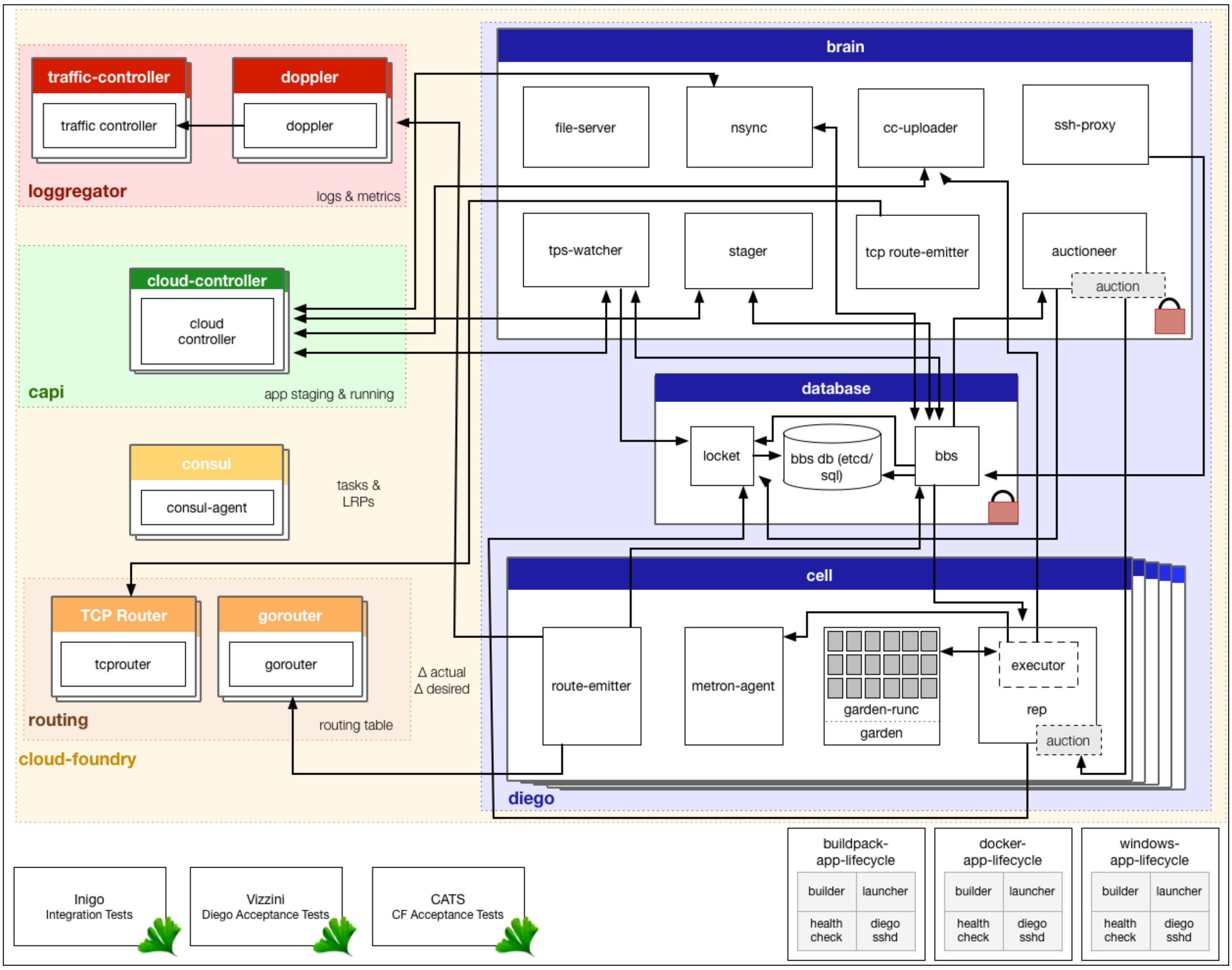
This topic includes the following sections:

- Architecture Diagram
- Diego Components
  - Diego Brain
  - Diego Cell
  - Database VMs
  - Consul
- Platform-specific Components
  - Garden Backends
  - App Lifecycle Binaries

Architecture Diagram

Cloud Foundry uses the Diego architecture to manage application containers. Diego components assume application scheduling and management responsibility from the Cloud Controller.

Refer to the following diagram and descriptions for information about the way Diego handles application requests.



View a larger version of this image.

1. The Cloud Controller passes requests to stage and run applications to several components on the Diego Brain.
2. The Diego Brain components translate staging and running requests into Tasks and Long Running Processes (LRPs), then submit these to the Bulletin Board System (BBS) through an API over HTTP.
3. The BBS submits the Tasks and LRPs to the Auctioneer, part of the Diego Brain.
4. The Auctioneer distributes these Tasks and LRPs to Cells through an Auction. The Diego Brain communicates with Diego Cells using SSL/TLS protocol.
5. Once the Auctioneer assigns a Task or LRP to a Cell, an in-process Executor creates a Garden container in the Cell. The Task or LRP runs in the container.
6. The BBS tracks desired LRPs, running LRP instances, and in-flight Tasks. It also periodically analyzes this information and corrects discrepancies to ensure consistency between `ActualLRP` and `DesiredLRP` counts.
7. The Metron Agent, part of the Cell, forwards application logs, errors, and metrics to the Cloud Foundry Loggregator. For more information, see the Application Logging in Cloud Foundry topic.

Diego Components

Diego components run and monitor Tasks and LRPs.

Diego Brain

Diego Brain components distribute Tasks and LRPs to Diego Cells, and correct discrepancies between `ActualLRP` and `DesiredLRP` counts to ensure fault-tolerance and long-term consistency.

The Diego Brain consists of the following:

- Auctioneer
- CC-Uploader
- File Server
- SSH Proxy
- TPS Watcher
- TCP Route-emitter
- Nsync
- Stager

Auctioneer

- Uses the auction package to run Diego Auctions for Tasks and LRPs
- Communicates with Cell Reps over SSL/TLS
- Maintains a lock in the BBS that restricts auctions to one Auctioneer at a time

Refer to the Auctioneer repository on GitHub for more information.

CC-Uploader

- Mediates uploads from the Executor to the Cloud Controller
- Translates simple HTTP POST requests from the Executor into complex multipart-form uploads for the Cloud Controller

Refer to the CC-Uploader repository on GitHub for more information.

File Server

- This "blobstore" serves static assets that can include general-purpose App Lifecycle binaries and application-specific droplets and build artifacts.

Refer to the File Server repository on GitHub for more information.

SSH Proxy

- Brokers connections between SSH clients and SSH servers running inside instance containers

Refer to Understanding Application SSH, Application SSH Overview, or the Diego SSH Github repository for more information.

TPS Watcher

- Provides the Cloud Controller with information about currently running LRPs to respond to `cf apps` and `cf app APP_NAME` requests
- Monitors `ActualLRP` activity for crashes and reports them the Cloud Controller

Refer to the TPS repository on GitHub for more information.

TCP Route-emitter

- Monitors `DesiredLRP` and `ActualLRP` states, emitting TCP route registration and unregistration messages to the Cloud Foundry routing API when it detects changes
- Periodically emits TCP routes to the Cloud Foundry routing API

Nsync

- Listens for app requests to update the `DesiredLRPs` count and updates `DesiredLRPs` through the BBS
- Periodically polls the Cloud Controller for each app to ensure that Diego maintains accurate `DesiredLRPs` counts

Refer to the Nsync repository on GitHub for more information.

Stager

- Translates staging requests from the Cloud Controller into generic Tasks and LRPs
- Sends a response to the Cloud Controller when a Task completes

Refer to the Stager repository on GitHub for more information.

Diego Cell

Diego Cell components manage and maintain Tasks and LRPs.

The Diego Cell consists of the following:

- Rep
- Executor
- Garden
- Metron Agent
- Route-emitter

Rep

- Represents a Cell in Diego Auctions for Tasks and LRPs
- Mediates all communication between the Cell and the BBS
- Ensures synchronization between the set of Tasks and LRPs in the BBS with the containers present on the Cell

- Maintains the presence of the Cell in the BBS
- Runs Tasks and LRPs by asking the in-process Executor to create a container and `RunAction` recipes

Refer to the Rep repository on GitHub for more information.

Executor

- Runs as a logical process inside the Rep
- Implements the generic Executor actions detailed in the API documentation
- Streams `STDOUT` and `STDERR` to the Metron agent running on the Cell

Refer to the Executor repository on GitHub for more information.

Garden

- Provides a platform-independent server and clients to manage Garden containers
- Defines the Garden-runC interface for container implementation

See the Garden topic or the Garden repository on GitHub for more information.

Metron Agent

Forwards application logs, errors, and application and Diego metrics to the Loggregator Doppler component

Refer to the Metron repository on GitHub for more information.

Route-emitter

- Monitors `DesiredLRP` and `ActualLRP` states, emitting route registration and unregistration messages to the Cloud Foundry Gorouter when it detects changes
- Periodically emits the entire routing table to the Cloud Foundry Gorouter

Refer to the Route-Emitter repository on GitHub for more information.

Database VMs

The Diego database VM consists of the following components.

Diego Bulletin Board System

- Maintains a real-time representation of the state of the Diego cluster, including all desired LRPs, running LRP instances, and in-flight Tasks
- Provides an RPC-style API over HTTP to Diego Core components and external clients, including the SSH Proxy and Route Emitter.
- Ensure consistency and fault tolerance for Tasks and LRPs by comparing desired state (stored in the database) with actual state (from running instances)
- Acts to keep `DesiredLRP` count and `ActualLRP` count synchronized in the following ways:
  - If the `DesiredLRP` count exceeds the `ActualLRP` count, requests a start auction from the Auctioneer
  - If the `ActualLRP` count exceeds the `DesiredLRP` count, sends a stop message to the Rep on the Cell hosting an instance
- Monitors for potentially missed messages, resending them if necessary

Refer to the Bulletin Board System repository on GitHub for more information.

MySQL

- Provides a consistent key-value data store to Diego

Locket

- Provides a consistent key-value store for maintenance of distributed locks and component presence

Go MySQL Driver

The Diego BBS stores data in MySQL. Diego uses the Go MySQL Driver to communicate with MySQL.

Refer to the Go MySQL Driver repository on GitHub for more information.

Consul

- Provides dynamic service registration and load balancing through DNS resolution

Refer to the Consul repository on GitHub for more information.

Platform-specific Components

Garden Backends

Garden contains a set of interfaces that each platform-specific backend must implement. See the Garden topic or the Garden repository on GitHub for more information.

App Lifecycle Binaries

The following three platform-specific binaries deploy applications and govern their lifecycle:

- The **Builder**, which stages a CF application. The Builder runs as a Task on every staging request. It performs static analysis on the application code and does any necessary pre-processing before the application is first run.
- The **Launcher**, which runs a CF application. The Launcher is set as the Action on the `DesiredLRP` for the application. It executes the start command with the correct system context, including working directory and environment variables.
- The **Healthcheck**, which performs a status check on running CF application from inside the container. The Healthcheck is set as the Monitor action on the `DesiredLRP` for the application.

Current Implementations

- Buildpack App Lifecycle implements the Cloud Foundry buildpack-based deployment strategy.
- Docker App Lifecycle implements a Docker deployment strategy.

Application SSH Components and Processes

Page last updated:

This document describes details about the Pivotal Application Service SSH components for access to deployed application instances. Pivotal Application Service supports native SSH access to applications and load balancing of SSH sessions with the load balancer for your PAS deployment.

The SSH Overview document describes procedural and configuration information about application SSH access.

SSH Components

The PAS SSH includes the following central components, which are described in more detail below:

- An implementation of an SSH proxy server.
- A lightweight SSH daemon.

If these components are deployed and configured correctly, they provide a simple and scalable way to access containers apps and other long running processes (LRPs).

SSH Daemon

The SSH daemon is a lightweight implementation that is built around the Go SSH library. It supports command execution, interactive shells, local port forwarding, and secure copy. The daemon is self-contained and has no dependencies on the container root file system.

The daemon is focused on delivering basic access to application instances in PAS. It is intended to run as an unprivileged process, and interactive shells and commands will run as the daemon user. The daemon only supports one authorized key, and it is not intended to support multiple users.

The daemon can be made available on a file server and Diego LRPs that want to use it can include a download action to acquire the binary and a run action to start it. PAS applications will download the daemon as part of the lifecycle bundle.

SSH Proxy Authentication

The SSH proxy hosts the user-accessible SSH endpoint and is responsible for authentication, policy enforcement, and access controls in the context of PAS. After a user has successfully authenticated with the proxy, the proxy will attempt to locate the target container and create an SSH session to a daemon running inside the container. After both sessions have been established, the proxy will manage the communication between the user’s SSH client and the container’s SSH Daemon.

How Diego Balances App Processes

Page last updated:

Diego balances app processes over the virtual machines (VMs) in a Cloud Foundry (CF) installation using the Diego Auction. When new processes need to be allocated to VMs, the Diego Auction determines which ones should run on which machines. The auction algorithm balances the load on VMs and optimizes app availability and resilience. This topic explains how the Diego Auction works.

Refer to the Auction repository on GitHub for source code and more information.

Tasks and Long-Running Processes

The Diego Auction distinguishes between two types of jobs: **Tasks** and **Long-Running Processes** (LRPs).

- **Tasks** run once, for a finite amount of time. A common example is a staging task that compiles an app's dependencies, to form a self-contained droplet that makes the app portable and runnable on multiple VMs. Other examples of tasks include making a database schema change, bulk importing data to initialize a database, and setting up a connected service.
- **Long-Running Processes** run continuously, for an indefinite amount of time. LRPs terminate only if stopped or killed, or if they crash. Examples include web servers, asynchronous background workers, and other applications and services that continuously accept and process input. To make high-demand LRPs more available, Diego may allocate multiple instances of the same application to run simultaneously on different VMs, often spread across Availability Zones that serve users in different geographic regions.

The Diego Auction process repeats whenever new jobs need to be allocated to VMs. Each auction distributes a current **batch** of work, Tasks and LRPs, that can include newly-created jobs, jobs left unallocated in the previous auction, and jobs left orphaned by failed VMs. Diego does not redistribute jobs that are already running on VMs. Only one auction can take place at a time, which prevents placement collisions.

Ordering the Auction Batch

The Diego Auction algorithm allocates jobs to VMs to fulfill the following outcomes, in decreasing **priority** order:

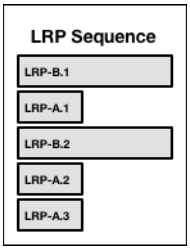
1. Keep at least one instance of each LRP running.
2. Run all of the Tasks in the current batch.
3. Distribute as much of the total desired LRP load as possible over the remaining available VMs, by spreading multiple LRP instances broadly across VMs and their Availability Zones.

To achieve these outcomes, each auction begins with the Diego Auctioneer component arranging the batch's jobs into a priority order. Some of these jobs may be duplicate instances of the same process that Diego needs to allocate for high-traffic LRPs, to meet demand. So the Auctioneer creates a list of multiple LRP instances based on the desired instance count configured for each process.

For example, if the process LRP-A has a desired instance count of 3 and a memory load of 2, and process LRP-B has 2 desired instances and a load of 5, the Auctioneer creates a list of jobs for each process as follows:

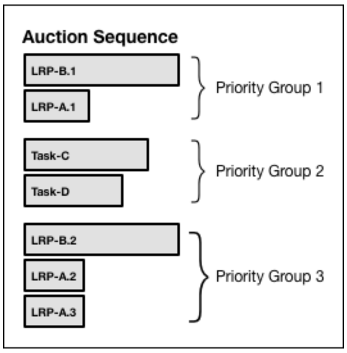
Process	Desired Instances	Load	Jobs
LRP-A	3	2	LRP-A.1 LRP-A.2 LRP-A.3
LRP-B	2	5	LRP-B.1 LRP-B.2

The Auctioneer then builds an ordered sequence of LRP instances by cycling through the list of LRPs in decreasing order of load. With each cycle, it adds another instance of each LRP to the sequence, until all desired instances of the LRP have been added. With the example above, the Auctioneer would order the LRPs like this:



The Auctioneer then builds an ordered sequence for all jobs, both LRPs and Tasks. Reflecting the auction batch priority order, the first instances of LRPs are first priority. Tasks are next, in decreasing order of load. Duplicate LRP jobs come last.

Adding one-time Task-C (load = 4) and Task-D (load = 3) to the above example, the priority order becomes:



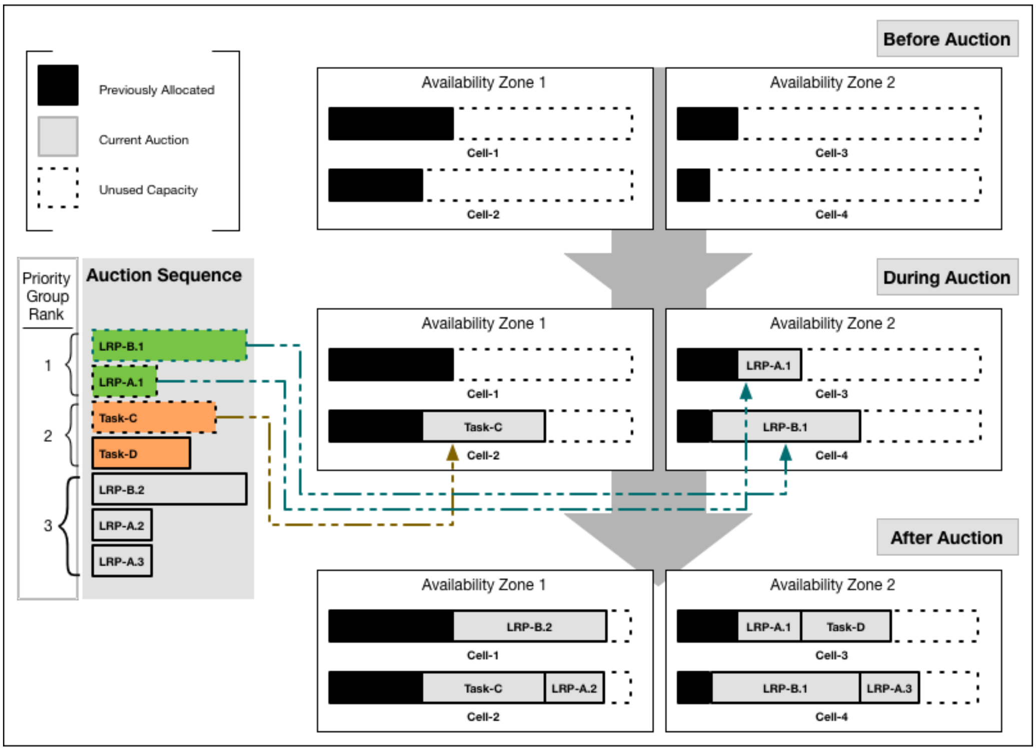
Auctioning the Batch to the Cells

With all jobs sorted in priority order, the Auctioneer allocates each in turn to one of the VMs. The process resembles an auction, where VMs "bid" with their suitability to run each job. Facilitating this process, each app VM has a resident Cell that monitors and allocates the machine's operation. The Cell participates in the auction on behalf of the virtual machine that it runs on.

Starting with the highest-priority job in the ordered sequence, the Auctioneer polls all the Cells on their fitness to run the currently-auctioned job. Cells "bid" to host each job according to the following priorities, in decreasing order:

1. Allocate all jobs only to Cells that have the correct software stack to host them, and sufficient resources given their allocation so far during this auction.
2. Allocate LRP instances into Availability Zones that are not already hosting other instances of the same LRP.
3. Within each Availability Zone, allocate LRP instances to run on Cells that are not already hosting other instances of the same LRP.
4. Allocate any job to the Cell that has lightest load, from both the current auction and jobs it has been running already. In other words, distribute the total load evenly across all Cells.

Our example auction sequence has seven jobs: five LRP instances and two Tasks. The following diagram shows how the Auctioneer might distribute this work across four Cells running in two Availability Zones:



If the Auctioneer reaches the end of its sequence of jobs, having distributed all jobs to the Cells, it submits requests to the Cells to execute their allotted work. If the Cells ran out of capacity to handle all jobs in the sequence, the Auctioneer carries the unallocated jobs over and merges them into the next auction batch, to be allocated in the next auction.

Triggering Another Auction

The Diego Auction process repeats to adapt a Cloud Foundry deployment to its changing workload. For example, the BBS initiates a new auction when it detects that the actual number of running instances of LRPs does not match the number desired. Diego's BBS component monitors the number of instances of each LRP that are currently running. The BBS component periodically compares this number with the desired number of LRP instances, as configured by the user. If the actual number falls short of what is desired, the BBS triggers a new auction. In the case of a surplus of application instances, the BBS kills the extra instances and initiates another auction.

The Cloud Controller also triggers an auction whenever a Cell fails. After any auction, if a Cell responds to its work request with a message that it cannot perform the work after all, the Auctioneer carries the unallocated work over into the next batch. But if the Cell fails to respond entirely, for example if its connection times out, the unresponsive Cell may still be running its work. In this case, the Auctioneer does not automatically carry the Cell's work over to the next batch. Instead, the Auctioneer defers to the BBS to continue monitoring the states of the Cells, and to re-assign unassigned work later if needed.

PCF Operator Guide

For PCF Operators

This guide shows you how to run a PCF platform. This ongoing responsibility may include but is not limited to:

- Configuring PCF capabilities
- Integrating PCF with external systems
- Updating PCF and installed products
- Monitoring PCF health and performance
- Adjusting PCF resources and options to fix health or performance issues
- Diagnosing and troubleshooting PCF problems
- Managing Pivotal Application Service (PAS) users, resources, and infrastructure
- Installing software services and otherwise enabling PCF developers
- Maintaining PCF
- Keeping PCF secure

If you do these things, you are a PCF **operator**, and the contents of this guide are for you.

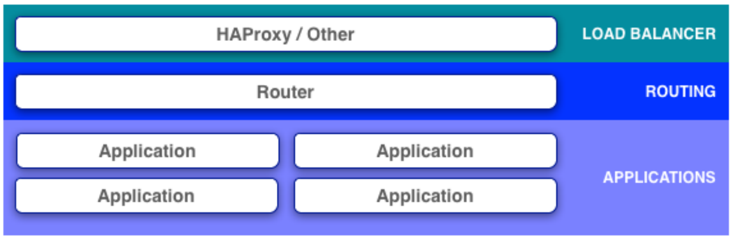
Guide Contents

- "Day 2" Configurations - Setting up internal operations and external integrations for PCF.
- Ongoing Operations - Routine procedures for running and growing the platform, including:
  - Upgrading
  - Monitoring, Logging, and Reporting
  - Platform Tuning
  - Enabling Developers
  - Backing Up
  - Security
- Managing App Runtimes - Procedures performed by people with administrator or manager roles in Pivotal Application Service (PAS), such as managing users, orgs, spaces, and service instances. Operators can perform these actions by logging in with Admin credentials, which grants them the role of Org Manager across all PAS orgs.
- Using Ops Manager - Ops Manager's dashboard interface streamlines the installation, configuration, and upgrading of PCF platform services and add-ons.
- Cloud Foundry Command Line Interface (cf CLI) - Using the cf CLI to send commands to the Cloud Controller, the executive component of PAS.
- Troubleshooting and Diagnostics - Tools and procedures for troubleshooting PCF.

Understanding the PAS Network Architecture

Page last updated:

The diagram below shows the key Pivotal Application Service (PAS) network components.



Load Balancer

PAS includes an HAProxy load balancer for terminating SSL. If you do not want to serve SSL certificates for PAS on your own load balancer use the HAProxy. If you do choose to manage SSL yourself, omit the HAProxy by setting the number of instances to zero in Ops Manager.

Router

The routers in PAS are responsible for routing HTTP requests from web clients to application instances in a load balanced fashion. The routers are dynamically configured based on users mapping of applications to location URLs called routes, and updated by the runtime service as application instances are dynamically distributed.

For high availability, the routers are designed to be horizontally scalable. Configure your load balancer to distribute incoming traffic across all router instances.

Refer to the Cloud Foundry Architecture topic for more information about Cloud Foundry components.

Identifying the API Endpoint for your PAS Instance

Page last updated:

The API endpoint for your Pivotal Application Service (PAS) deployment, its target URL, is the API endpoint of the deployment's Cloud Controller. Find your Cloud Controller API endpoint by consulting your cloud operator, from the Apps Manager, or from the command line.

From the Apps Manager

Log in to the Apps Manager for your PAS instance, then click **Tools** in the left navigation panel. The **Getting Started** section of the Tools page shows your API endpoint.

From the Command Line

From a command line, use the `cf api` command to view your API endpoint.

Example:

```
$ cf api
API endpoint: https://api.example.com (API version: 2.2.0)
```

Configuring SSL/TLS Termination at HAProxy

Page last updated:

Both Pivotal Application Service (PAS) and Isolation Segments for Pivotal Cloud Foundry include an HAProxy instance.

HAProxy is appropriate to use in a deployment when features are needed that are offered by HAProxy but are not offered by the CF Routers or IaaS-provided load balancers such as with Azure load balancers. These include filtering of protected domains from trusted networks.

While HAProxy instances provide load balancing for the Gorouters, HAProxy is not itself highly available. For production environments, use a highly-available load balancer to scale HAProxy horizontally. The load balancer does not need to terminate TLS or even operate at layer 7 (HTTP); it can simply provide layer 4 load balancing of TCP connections. Use of HAProxy does not remove the need for CF Routers; the Gorouter must always be deployed for HTTP applications, and TCP Router for non-HTTP applications.

You can generate a self-signed certificate for HAProxy if you do not want to obtain a signed certificate from a well-known certificate authority.

Procedure: Terminate SSL/TLS at HAProxy

In PCF, perform the following steps to configure SSL termination on HAProxy:

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the **Pivotal Application Service** tile in the Installation Dashboard.
3. Click **Networking**.
4. Configure the following based on the IaaS of your PCF deployment.

If your PCF deployment is on:	Then configure the following:	See also:
OpenStack or vSphere	Decide whether you want your HAProxy to be highly available. ◦ If you need highly available HAProxy, then perform the following steps: 1. Choose an IP address for each HAProxy instance on the subnet where you deployed PCF. 2. In the **HAProxy IPs** field of the **Networking** page, enter the IP addresses you have selected for your HAProxy instances. 3. Configure your load balancer (for example, F5 or NSX) to forward domain names to the HAProxy IP addresses. ◦ If you do not require high availability (for example, you are setting up a development environment), then perform the following steps: 1. Skip setting up the load balancer. 2. Choose one IP address for the single HAProxy instance. 3. Configure DNS to point at the IP address. See How to Set Up DNS for HAProxy.	For more information, see the Pivotal Application Service (PAS) networking configuration topic for OpenStack or vSphere.
AWS, GCP or Azure	1. Leave the HAProxy IP address blank. 2. In the **Resource Config** page of PAS tile, locate the HAProxy job. 3. In the **Load Balancer** column for the HAProxy job, specify the appropriate IaaS load balancer resource.	For more information, see the PAS installation instructions for AWS, Azure, or GCP.

5. In the **Certificates and Private Keys for HAProxy and Router** field, click the **Add** button to define at least one certificate keypair for HAProxy and Router. For each certificate keypair that you add, assign a name, enter the PEM-encoded certificate chain and PEM-encoded private key. You can either upload your own certificate or generate an RSA certificate in PAS. For options and instructions on creating a certificate for your wildcard domains, see Creating a Wildcard Certificate for PCF Deployments.
6. In the **Minimum version of TLS supported by HAProxy and Router**, select the minimum version of TLS to use in HAProxy communications. HAProxy

use TLS v1.2 by default. If you need to accommodate clients that use an older version of TLS, select a lower minimum version. For a list of TLS ciphers supported by the HAProxy, see TLS Cipher Suites.

7. Under **HAProxy forwards requests to Router over TLS**, leave **Enabled** selected and provide the backend certificate authority.

8. If you want to use a specific set of TLS ciphers for HAProxy, configure **TLS Cipher Suites for HAProxy**. Enter an ordered, colon-separated list of TLS cipher suites in the OpenSSL format. For example, if you have selected support for an earlier version of TLS, you can enter cipher suites supported by this version. For a list of TLS ciphers supported by the HAProxy, see TLS Cipher Suites.

9. If you expect requests larger than the default maximum of 16 Kbytes, enter a new value (in bytes) for **HAProxy Request Max Buffer Size**. You may need to do this, for example, to support apps that embed a large cookie or query string values in headers.

10. (Optional) If you are not using SSL encryption or if you are using self-signed certificates, you can select **Disable SSL certificate verification for this environment**. Selecting this checkbox also disables SSL verification for route services.

    Use this checkbox only for development and testing environments. Do not select it for production environments.

11. (Optional) If you do not want HAProxy or the Gorouter to accept any non-encrypted HTTP traffic, select the **Disable HTTP on HAProxy and Router** checkbox.

12. In the **Configure the CF Router support for the X-Forwarded-Client-Cert header** field, select **Always forward the XFCC header in the request, regardless of the whether the client connection is mTLS**.

13. (Optional) If your PCF deployment uses HAProxy and you want it to receive traffic only from specific sources, use the following fields:
    - **Protected Domains**: Enter a comma-separated list of domains from which PCF can receive traffic.
    - **Trusted CIDRs**: Optionally, enter a space-separated list of CIDRs to limit which IP addresses from the Protected Domains can send traffic to PCF.

14. Click **Save**.

How to Set Up DNS for HAProxy

You only need to perform this procedure if you are using one instance of HAProxy such as in a development environment. If you would like HAProxy to be highly available, you must have a load balancer in front of it. In this case, you would point DNS at the load balancer.

To use a single instance HAProxy load balancer in a vSphere or OpenStack deployment, create a wildcard A record in your DNS and configure some fields in the PAS product tile.

1. Create an A record in your DNS that points to the HAProxy IP address. The A record associates the **System Domain** and **Apps Domain** that you configure in the **Domains** section of the PAS tile with the HAProxy IP address.

   For example, with `cf.example.com` as the main subdomain for your Cloud Foundry (CF) deployment and an HAProxy IP address `203.0.113.1`, you must create an A record in your DNS that serves `example.com` and points `*.cf` to `203.0.113.1`.

Name	Type	Data	Domain
*.cf	A	203.0.113.1	example.com

2. Use the Linux `host` command to test your DNS entry. The `host` command should return your HAProxy IP address.

   Example:

   ```
   $ host cf.example.com
   cf.example.com has address 203.0.113.1
   $ host anything.example.com
   anything.cf.example.com has address 203.0.113.1
   ```

Configuring Proxy Settings for All Applications

This topic describes how to globally configure proxy settings for all applications in your Pivotal Cloud Foundry (PCF) deployment. Some environments restrict access to the Internet by requiring traffic to pass through an HTTP or HTTPS proxy. PCF operators can use the Cloud Foundry Command Line Interface (cf CLI) to provide the proxy settings to all applications, including system applications and service brokers.

> **Note**: Incorrectly configuring proxy settings can prevent applications from connecting to the Internet or accessing required resources. They can also cause errands to fail and break system applications and service brokers. Although errands, system applications, and service brokers do not need to connect to the Internet, they often need to access other resources on PCF. Incorrect proxy settings can break these connections.

Set Environment Variables

To globally configure proxy settings for PCF applications, perform the following steps to set three environment variables for both the staging environment variable group and the running environment variable group.

For more information about variable groups, see the Environment Variable Groups section in the *Cloud Foundry Environment Variables* topic.

This procedure explains how to set proxy information for both staging and running applications. However, you can set proxy settings for only staging or only running applications.

1. Target your Cloud Controller with the cf CLI. If you have not installed the cf CLI, see the Installing the cf CLI topic.

```
$ cf api api.YOUR-SYSTEM-DOMAIN
Setting api endpoint to api.YOUR-SYSTEM-DOMAIN...
OK
API endpoint:   https://api.YOUR-SYSTEM-DOMAIN (API version: 2.54.0)
Not logged in. Use 'cf login' to log in.
```

2. Log in with your UAA administrator credentials. To retrieve these credentials, navigate to the Pivotal Application Service (PAS) tile in the Ops Manager Installation Dashboard and click **Credentials**. Under **UAA**, click **Link to Credential** next to **Admin Credentials** and record the password.

```
$ cf login
API endpoint: https://api.YOUR-SYSTEM-DOMAIN


Email> admin
Password>
Authenticating...
OK
```

3. To configure proxy access for applications that are staging, run the following command, replacing the placeholder values:

```
$ cf set-staging-environment-variable-group '{"http_proxy": "http://YOUR-PROXY:8080/", "https_proxy": "http://YOUR-PROXY:8080/", "no_proxy": "NO-PROXY.EXAMPLE.COM"
```

   - `http_proxy`: Set this value to the proxy to use for HTTP requests.
   - `https_proxy`: Set this value to the proxy to use for HTTPS requests. In most cases, this will be the same as `http_proxy`.
   - `no_proxy`: Set this value to a comma-separated list of DNS names or IP addresses that can be accessed without passing through the proxy. This value may not be needed, because it depends on your proxy configuration. From now on, the proxy settings are applied to staging applications.

4. To configure proxy access for applications that are running, run the following command, replacing the placeholder values as above:

```
$ cf set-running-environment-variable-group '{"http_proxy": "http://YOUR-PROXY:8080/", "https_proxy": "http://YOUR-PROXY:8080/", "no_proxy": "NO-PROXY.EXAMPLE.COM"
```

   To configure proxy settings for Java-based applications, use the following command instead, replacing the placeholder values. For `http.nonProxyHosts`, use a pipe-delimited list rather than a comma-separated list.

```
$ cf set-running-environment-variable-group '{"JAVA_OPTS": "-Dhttp.proxyHost=YOUR-PROXY -Dhttp.proxyPort=8080 -Dhttp.nonProxyHosts=NO-PROXY.EXAMPLE.COM"}'
```

   For more information about these Java proxy settings, see Java Networking and Proxies.

5. To apply the proxy configuration for the running environment variable group, you must restart each application that you want to use the new configuration.

Troubleshooting

If an application fails after you apply the global proxy settings, try the following solutions.

Exclude an App From Global Proxy Settings

If your application fails, try instructing the application to ignore the global proxy settings. Perform the following commands to manually unset the proxy environment variables for the failing application:

1. Set the proxy environment variables for `http_proxy` to an empty value:

   ```
   $ cf set-env YOUR-APP http_proxy ''
   ```

2. Set the proxy environment variables for `https_proxy` to an empty value:

   ```
   $ cf set-env YOUR-APP https_proxy ''
   ```

3. Set the proxy environment variables for `no_proxy` to an empty value:

   ```
   $ cf set-env YOUR-APP no_proxy ''
   ```

Change Case of HTTP

Your application and language runtime may be case-sensitive. Try performing the steps in the Set Environment Variables section using uppercase for `HTTP_PROXY`, `HTTPS_PROXY`, and `NO_PROXY` instead of lowercase. Refer to the following example.

```
$ cf set-staging-environment-variable-group '{"HTTP_PROXY": "http://YOUR-PROXY:8080/", "HTTPS_PROXY": "http://YOUR-PROXY:8080/"}'.
```

Check Proxy Settings

If you have set up your proxy so that it can only send traffic to the Internet, then a request to an internal resource like PCF fails. You must set `no_proxy` so that traffic destined for PCF and other internal resources is sent directly and does not go through the proxy. For instance, setting `no_proxy` to include your system and application domains will ensure that requests destined for those domains are sent directly.

Verify Interpretation

The interpretation of `no_proxy` depends on the application and the language runtime. Most support `no_proxy`, but the specific implementation may vary. For example, some match DNS names that end with the value set in `no_proxy`: `example.com` would match `test.example.com`. Others support the use of the asterisk as a wildcard to provide basic pattern matching in DNS names: `*.example.com` would match `test.example.com`. Most applications and language runtimes do not support pattern matching and wildcards for IP addresses.

Restricting App Access to Internal PCF Components

This topic describes how to secure the component virtual machines (VMs) of your Pivotal Cloud Foundry (PCF) deployment from being accessed by apps.

Introduction

See the following list to understand the concepts for this topic:

- **How PCF determines where apps can send traffic**:
  - PCF uses *Application Security Groups (ASGs)*, which are network policy rules specifying protocols, ports, and IP ranges that apply to outbound network connections initiated from apps. See Understanding ASGs.
- **Why you must create new rules for outbound app traffic**:
  - PCF installs with a default ASG that allows apps running on your deployment to send traffic to almost any IP address. This means apps are not blocked from initiating connections to most network destinations unless an administrator takes action to update the ASGs with a more restrictive policy.
- **How you can set up new rules**:
  - To help secure your component VMs against apps while ensuring your apps can access the services they need, follow the procedure below, which includes these steps:

Step	Description
1	Determine Your Network Layout: The procedure for securing your deployment with ASGs varies depending on your network layout, which you can determine using Ops Manager.
2	Ensure Access for PCF System Apps: Bind the default ASG to the `system` org so that PCF system apps can continue accessing the system components they need after you remove the deployment-wide default ASG in Step 4.
3	Create New ASGs: Block apps from sending traffic to system components, but allow them to send traffic to the services they need.
4	Remove the Default ASG: After you create and bind new ASGs, you no longer need the deployment-wide default ASG bindings that allow apps to send traffic to any IP.
5	Restart your Apps: To apply the ASG changes, you must restart all of the apps in your deployment.

- **When to set up new rules**:
  - Pivotal recommends that you complete this procedure directly after installing PCF, prior to developers pushing apps to the platform. If you complete the procedure after apps have been pushed to the platform, you must restart all the apps in your deployment.

Prerequisites

The procedure below requires that you have the latest release of ASG Creator from the Cloud Foundry incubator repository on Github. See About the ASG Creator Tool.

Procedure

Follow these steps to apply ASGs that prevent apps running on your deployment from accessing internal PCF components.

Step 1: Determine Your Network Layout

The procedure for securing your deployment with ASGs varies depending on your network layout, which you can determine by following these steps:

1. Log in to Ops Manager.

2. For each tile, click **Assign AZs and Networks** and record the selected **Network** that the tile is installed on.

3. Based on the information you gathered, determine which of the following network layouts you have:

Layout Name	Layout Description
One Network	◦ One network for Ops Manager and the Ops Manager Director, Pivotal Application Service (PAS), and services. **Note**: You cannot secure your deployment with ASGs if you have this network layout. Because PCF dynamically allocates IPs, they cannot be easily excluded in the case of a single network.
Two Networks	◦ One network for Ops Manager and the Ops Manager Director. ◦ One network for PAS and Services.
Three Networks	◦ One network for Ops Manager and the Ops Manager Director. ◦ One network for PAS. ◦ One network for all services.
Three or More Networks	◦ One network for Ops Manager and the Ops Manager Director. ◦ One network for PAS. ◦ One network for each service.

4. If your network layout includes two or more networks, continue Step 2: Ensure Access for PCF System Apps.

Step 2: Ensure Access for PCF System Apps

Follow these steps to apply the default ASG to the `system` org. This provides network access to PCF system apps without restrictions, which enables them to continue functioning properly after you perform Step 4: Remove the Deployment-wide Default ASG Binding.

1. Bind the default ASG to the staging set in the `system` org:

```
$ cf bind-staging-security-group default_security_group
```

2. Bind the default ASG to the running set in the `system` org:

```
$ cf bind-running-security-group default_security_group
```

Step 3: Create New ASGs

Follow these steps to create ASGs that block apps from accessing PCF components and create any additional ASGs that allow apps to access the services they require.

Part A: Record CIDRs

Gather the CIDRs for each network in your deployment:

1. From the Ops Manager Director tile, click **Create Networks** within the **Settings** tab.

2. In the **Networks** section, expand each network in your deployment by clicking its name.

3. Record the **CIDR** for each network.

Part B: Create and Bind ASGs that Block Network Access

Create ASGs that block apps from sending traffic to the networks that host Ops Manager, PAS, and, optionally, any installed services.

1. Create a `config.yml` containing the appropriate content for your network layout and replace the indicated values with the CIDRs you gathered:

- **Two Network Layout**:

```
excluded_networks:
- YOUR-OPS-MANAGER-CIDR
- YOUR-PAS-AND-SERVICES-CIDR
```

- **Three Network Layout**:

> **Note**: If you only want to secure the Ops Manager and PAS components, you can optionally exclude the services CIDR.

```
excluded_networks:
- YOUR-OPS-MANAGER-CIDR
- YOUR-PAS-CIDR
- YOUR-SERVICES-CIDR
```

- **Three or More Network Layout**:

> **Note**: If you only want to secure the Ops Manager and PAS components, you can optionally exclude the services CIDRs.

```
excluded_networks:
- YOUR-OPS-MANAGER-CIDR
- YOUR-PAS-CIDR
- YOUR-SERVICE-CIDR-1
- YOUR-SERVICE-CIDR-2
etc...
```

2. Run the following command to create a `json` that contains ASG rules, using your `config.yml` as input:

```
$ asg-creator create --config config.yml --output OUTPUT-FILE-NAME.json
```

Replace `OUTPUT-FILE-NAME` with a name of your choice.

3. Create an ASG by running the following command:

   a. Replace `SECURITY-GROUP-NAME` with a name of your choice.
   b. Replace `OUTPUT-FILE-NAME` with the name of the generated file from the previous step.

```
$ create-security-group SECURITY-GROUP-NAME OUTPUT-FILE-NAME.json
```

4. Bind the ASG to the default staging set:

```
$ cf bind-staging-security-group SECURITY-GROUP-NAME
```

5. Bind the ASG to the default running set:

```
$ cf bind-running-security-group SECURITY-GROUP-NAME
```

Part C: Create and Bind ASGs for Service Access

> **Note**: This part is only necessary if you blocked apps from a network that hosts services in the previous part. If you did not block apps from a network that hosts services, proceed to Step 4: Remove the Default ASG.

> **WARNING**: In the two network layout, PAS and services share the same network. This means that each time you create an ASG that allows apps to access a new port and protocol within the network, you further expose the PAS component VMs. This is a limitation of a two network layout. For guidance on network topology, see Reference Architectures.

Now that you have created ASGs to secure the Ops Man, PAS, and service components, work with developers to create additional ASGs that give apps access to the services they need.

For example, in any space where apps need to access the MySQL for PCF service, follow the steps in Creating Application Security Groups for MySQL.

For more information about creating and binding ASGs, see the following:

- Managing ASGs with the cf CLI
- Typical ASGs

Step 4: Remove the Default ASG

Now that you have bound new ASGs to determine outbound traffic rules, you no longer need the default ASG bindings that allow apps to send traffic to any IP address.

1. Unbind the default ASG from the staging set:

   ```
   $ cf unbind-staging-security-group default_security_group
   ```

2. Unbind the default ASG from the running set:

   ```
   $ cf unbind-running-security-group default_security_group
   ```

Step 5: Restart your Apps

To apply the ASG changes, you must restart all of the apps in your deployment. To mitigate app downtime during the restart, Pivotal recommends a blue-green deployment strategy.

> **Notes**: You do not need to restart the apps in the `system` org.

1. Work with developers to restart a few of their apps individually and test that they still work correctly with the new ASGs in place. If an app does not work as expected, you likely must create another ASG that allows the app to send traffic to a service it requires.

   > **Note**: To quickly roll back to the original overly-permissive state, you can re-bind the `default_security_group` ASG to the `default-staging` and `default-running` sets. You must then restart your apps to re-apply the original ASGs.

2. Restart the rest of the apps running on your deployment. Optionally, you can use the app-restarter cf CLI plugin to restart all apps in a particular space, org, or deployment.

Configuring Application Security Groups for Email Notifications

Page last updated:

To allow the Notifications Service to have network access you need to create Application Security Groups (ASGs).

Note: Without Application Security Groups the service is not usable.

Prerequisite

Review the Getting Started with the Notifications Service topic to ensure you have setup the service.

Configure Network Connections

The Notifications Service is deployed as a suite of applications to the `notifications-with-ui` space in the `system` org, and requires the following outbound network connections:

Destination	Ports	Protocol	Reason
`SMTP_SERVER`	587 (default)	tcp (default)	This service is used to send out email notifications
`LOAD_BALANCER_IP`	80, 443	tcp	This service will access the load balancer
`ASSIGNED_NETWORK`	3306	tcp	This service requires access to internal services. `ASSIGNED_NETWORK` is the CIDR of the network assigned to this service.

Note: The SMTP Server port and protocol are dependent on how you configure your server.

Create a SMTP Server ASG

1. Navigate to the Ops Manager **Installation Dashboard** and click the **Pivotal Application Service** tile > **Settings** tab.
2. Record the information in the **Address of SMTP Server** and **Port of SMTP Server** fields.
3. Using the **Address of SMTP Server** information you obtained in the previous step, find the IP addresses and protocol of your SMTP Server from the service you are using. You might need to contact your service provider for this information.
4. Create a `smtp-server.json` file. For `destination`, you must enter the IP address of your SMTP Server.

```
[
    {
      "protocol": "tcp",
      "destination": SMTP_SERVER_IPS,
      "ports": "587"
    }
]
```

5. Create a security group called `smtp-server`:

```
cf create-security-group smtp-server smtp-server.json
```

Create a Load Balancer ASG

Note: If you already have a ASG setup for a Load Balancer, you do not need to perform this step. Review your ASGs to check which groups you have setup.

If you are using the built-in HAProxy as your load balancer, follow this procedure. If you are using an external load balancer, you must obtain your

HAProxy IPs from the service you are using.

1. Record the **HAProxy IPs** in the **Pivotal Application Service** tile > **Settings** > **Networking** tab.

2. Create a `load-balancer-https.json` file. For `destination`, use the **HAProxy IPs** you recorded above.

```
[
  {
    "protocol": "tcp",
    "destination": "10.68.196.250",
    "ports": "80,443"
  }
]
```

3. Create a security group called `load-balancer-https`:

```
$ cf create-security-group load-balancer-https load-balancer-https.json
```

Create an Assigned Network ASG

Note: If you use external services, the IP addresses, ports, and protocols depend on the service.

1. Navigate to the Ops Manager **Installation Dashboard** > **Pivotal Application Service** tile > **Settings** > **Assign AZs and Networks** section.

2. Navigate to the network selected in the dropdown.

3. Record the **Ops Manager Director** tile > **Settings** tab > **Create Networks** > **CIDR** for the network identified in the previous step. Ensure the subnet mask allows the space to access `p-mysql`, `p-rabbitmq`, and `p-redis`.

4. Create a file `assigned-network.json`. For the `destination`, enter the **CIDR** you recorded above.

```
[
  {
    "protocol": "tcp",
    "destination": "10.68.0.0/20",
    "ports": "3306,5672,6379"
  }
]
```

5. Create a security group called `assigned-network`:

```
$ cf create-security-group assigned-network assigned-network.json
```

Bind the ASGs

1. Target the `system` org:

```
$ cf target -o system
```

2. Create a `notifications-with-ui` space:

```
$ cf create-space notifications-with-ui
```

3. Bind the ASGs you created in this topic to the `notifications-with-ui` space:

```
$ cf bind-security-group smtp-server system notifications-with-ui
$ cf bind-security-group load-balancer-https system notifications-with-ui
$ cf bind-security-group assigned-network system notifications-with-ui
```

Configuring SSH Access for PCF

Page last updated:

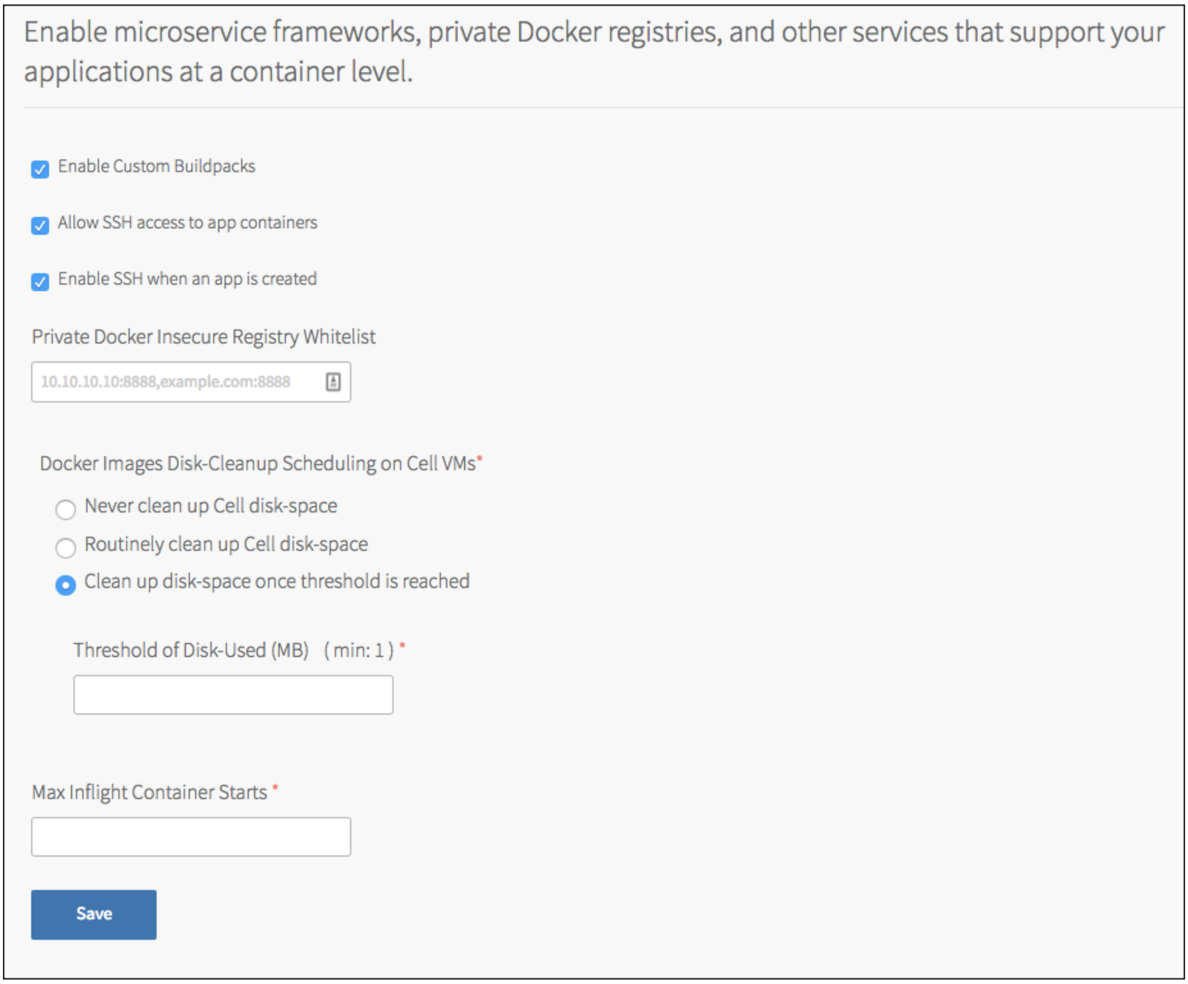
To help troubleshoot applications hosted by a deployment, Pivotal Cloud Foundry (PCF) supports SSH access into running applications. This document describes how to configure a PCF deployment to allow SSH access to application instances, and how to configure load balancing for those application SSH sessions.

Pivotal Application Service Configuration

This section describes how to configure Pivotal Application Service (PAS) to enable or disable deployment-wide SSH access to application instances. In addition to this deployment-wide configuration, Space Managers have SSH access control over their Space, and Space Developers have SSH access control over their to their Applications. For details about SSH access permissions, see the Application SSH Overview topic.

To configure PAS SSH access for application instances:

1. Open the PAS tile in Ops Manager.
2. Under the **Settings** tab, select the **Application Containers** section.
3. Enable or disable the **Allow SSH access to app containers** checkbox.
4. Optionally, select **Enable SSH when an app is created** to enable SSH access for new apps by default in spaces that allow SSH. If you deselect this checkbox, developers can still enable SSH after pushing their apps by running `cf enable-ssh APP-NAME`.



SSH Load Balancer Configuration

If you use HAProxy as a load balancer and SSH access is enabled, SSH requests are load balanced by HAProxy. This configuration relies on the presence of the same Consul server cluster that Diego components use for service discovery. This configuration also works well for deployments where all traffic on the system domain and its subdomains is directed towards the HAProxy job, as is the case for a BOSH-Lite Cloud Foundry deployment on the default `192.0.2.34.xip.io` domain.

For AWS deployments, where the infrastructure offers load-balancing as a service through ELBs, the deployment operator can provision an ELB to balance load across the SSH proxy instances. You should configure this ELB to listen to TCP traffic on the port given in `app_ssh.port` and to send it to port 2222.

To register the SSH proxies with this ELB, add the ELB identifier to the `elbs` property in the `cloud_properties` hash of the Diego manifest `access_zN` resource pools. If you used the Spiff-based manifest-generation templates to produce the Diego manifest, specify these `cloud_properties` hashes in the `iaas_settings.resource_pool_cloud_properties` section of the `iaas-settings.yml` stub.

Securing Services Instance Credentials with PAS CredHub (Beta)

This topic describes how Pivotal Cloud Foundry (PCF) operators can ensure service instance credentials are securely stored in CredHub.

Note: This feature is currently in Beta.

- **What is PAS CredHub?**
  - The Pivotal Application Service (PAS) tile includes its own CredHub component, separate from the CredHub component included with the Ops Manager Director tile. For more information about this centralized credential management component, see the CredHub documentation.
- **What is PAS CredHub used for?**
  - PAS CredHub exists to securely store service instance credentials. Previously, PCF could only use the Cloud Controller database for storing these credentials.
- **What are service instance credentials**?
  - When developers want their app to use a service, such as those provided by the Spring Cloud Services tile for PCF, they must bind their app to an instance of that service. Service bindings include credentials that developers can use to access the service from their app. For more information, see Binding Credentials.
- **How can I ensure that service instance credentials are stored in PAS CredHub?**
  - You must configure PAS to enable this functionality by following the procedure below. Not all services support the use of PAS CredHub.
- **Can I use PAS CredHub to store service instance credentials if some of my services do not support the use of PAS CredHub?**
  - PAS supports both services that do and do not use PAS CredHub. Services that do not use PAS CredHub continue to pass their credentials to the Cloud Controller database.
- **Can I rotate service instance credentials in PAS CredHub?**
  - PAS CredHub supports credential rotation. For more information, see the Rotating PAS CredHub Encryption Keystopic.

Prerequisite

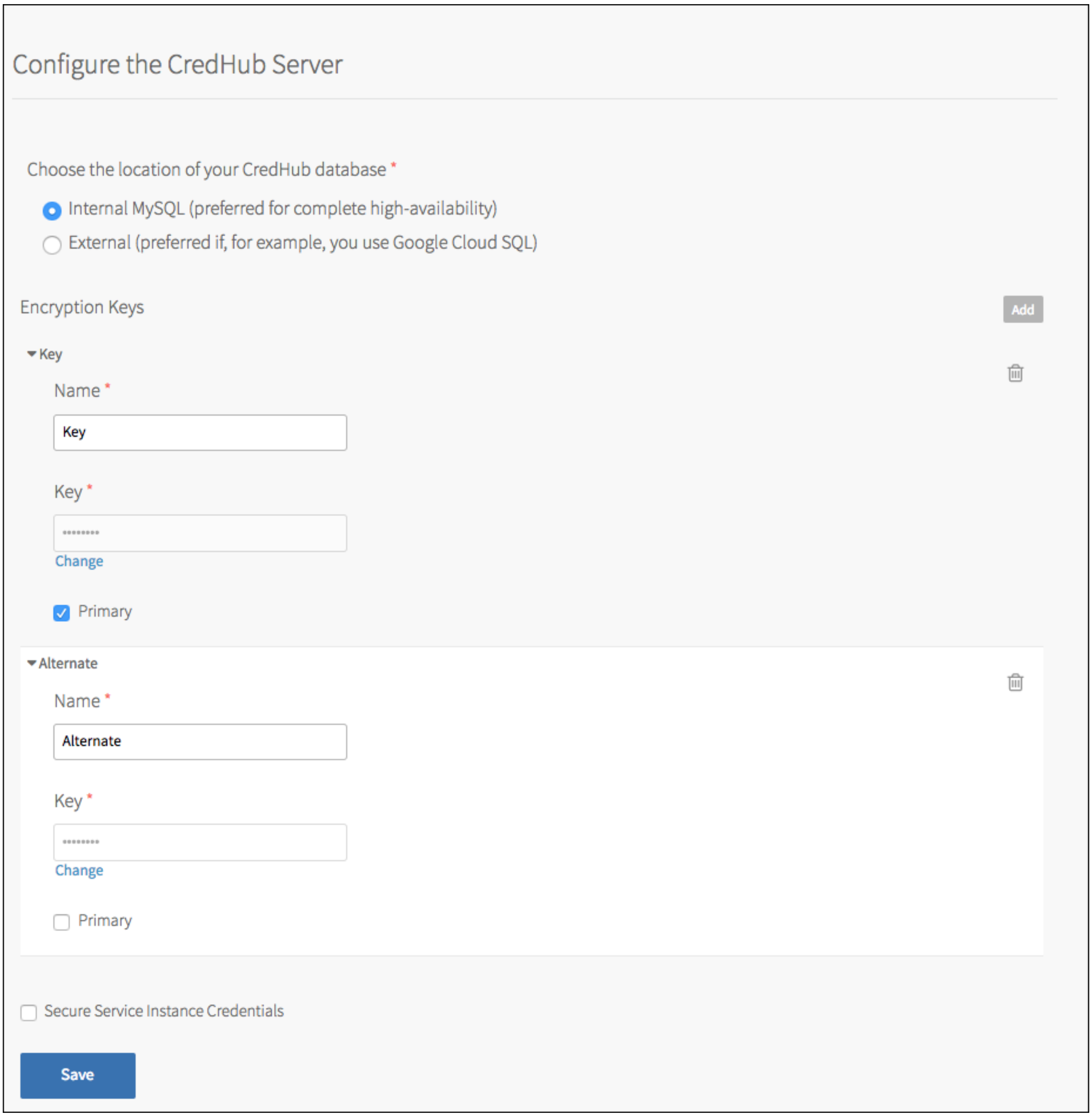
The procedures in this document are only effective for services that support storing their instance credentials in PAS CredHub. To learn whether a service supports this feature, see the documentation for that service.

Breaking Change: If you opt out of the BOSH DNS feature, your PCF deployment cannot support Secure Service Instance Credentials.

Step 1: Configure the PAS Tile

You must configure the PAS tile to support securing service instance credentials in CredHub. From the PAS tile in Ops Manager, complete the following steps:

1. Select the **CredHub** pane.



2. Choose the location of your CredHub Database. PAS includes this CredHub database for services to store their service instance credentials.

   a. If you chose **External**, enter the following:
      - **Hostname**: The IP address of your database server.
      - **TCP Port**: The port of your database server, such as `3306`.
      - **Username**: A unique username that can access this specific database on the database server.
      - **Password**: The password for the provided username.
      - **Database CA Certificate**: A certificate to use for encrypting traffic to and from the database.

3. Under **Encryption Keys**, specify a key to use for encrypting and decrypting the values stored in the CredHub database.
   - **Name**: Enter the name of the key.
   - **Key**: Enter a key that is at least 20 characters in length.
   - **Primary**: Select this checkbox to use this key as your primary key.

   > **Note**: Ensure that you only mark one key as **Primary**. The UI includes an **Add** button to add more keys to support key rotation.

4. If your deployment uses any PCF services that support storing service instance credentials in CredHub and you want to enable this feature, select the **Secure Service Instance Credentials** checkbox.

5. Select the **Resource Config** pane.

6. In the **CredHub** row, under the **Job** column, set the number of instances to `2`. This is the minimum instance count required for high availability.

Step 2: Create Application Security Groups

Application Security Groups (ASGs) are network policy rules specifying protocols, ports, and IP ranges that apply to outbound network connections initiated from apps. You must follow the steps below to ensure the ASGs for your deployment allow apps to communicate with the PAS CredHub API.

Note: The default ASGs in PCF allow apps running on your deployment to send traffic to almost any IP address. This step is only required if your ASGs restrict network access from apps to the network that PAS runs on, as documented in Restricting App Access to Internal PCF Components.

1. From the PAS tile, click **Assign AZs and Networks** and record the selected Network where the tile is installed.

2. From the Ops Manager Director tile, within the **Settings** tab, click **Create Networks**.

3. In the **Networks** section, click the name of the PAS network to expand it.

4. Record the CIDR for the PAS network.

5. Create a file named `pas-credhub.json` for specifying your ASG rules. Copy the content below into the file. Replace `YOUR-PAS-CIDR` with the CIDR you recorded in the previous step.

```
[
  {
  "protocol": "tcp",
  "destination": "YOUR-PAS-CIDR",
  "ports": "8844"
  }
]
```

6. Run the following command to create an ASG that allows apps to access the CredHub API:

```
$ cf create-security-group runtime-credhub ~/workspace/runtime-credhub runtime-credhub.json
```

7. Bind this ASG to your deployment or the specific space in which you want apps to access CredHub. For more information about binding ASGs, see Bind ASGs. Ensure that apps deployed as part of the service tile installation process have access to CredHub in addition to the apps pushed to the platform by developers. For example, the Spring Cloud Services tile deploys the `spring-cloud-broker` app to the `p-spring-cloud-services` space of the `systsem` org.

8. Restart apps for the ASGs to take effect. Optionally, you can use the app-restarter cf CLI plugin to restart all apps in a particular space, org, or deployment.

Step 3: Unbind and Rebind Service Instances

For any service instance bindings that existed before PAS CredHub was supported for that service, you must work with your developers to unbind and rebind the service instances to their apps. If you do not unbind and rebind the service, apps continue functioning as normal and fetching credentials from the Cloud Controller database.

Note: This step is not required for bindings created after you installed the new version of the service tile that supports CredHub and you completed the procedures in steps 1 and 2 of this topic.

1. Unbind the service instance from the app:

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

2. Rebind the service instance to the app:

```
$ cf bind-service YOUR-APP YOUR-SERVICE-INSTANCE
```

3. Review the `VCAP_SERVICES` environment variable to verify that the new service instance binding includes CredHub pointers:

```
$ cf env YOUR-APP
```

See VCAP_SERVICES for help parsing the output of the `cf-env` command.

4. Restart the app to apply the service instance binding:


```
cf restart YOUR-APP
```

If you run `cf-env` again, you can see the `VCAP-SERVICES` environment variable now contains the credentials for the service instance binding.

Identifying PAS Jobs Using vCenter

Page last updated:

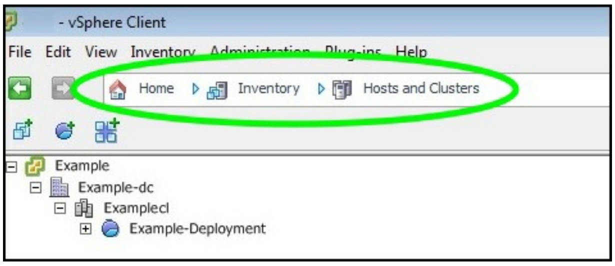
To effectively monitor, control, and manage the virtual machines making up your Pivotal Application Service (PAS) deployment, you may need to identify which VM corresponds to a particular job in PAS. You can find the CID of a particular VM from Pivotal Cloud Foundry (PCF) Operations Manager by navigating to PAS **Status**.

If you have deployed PAS to VMware vSphere, you can also identify which PAS job corresponds to which VM using the vCenter vSphere client.

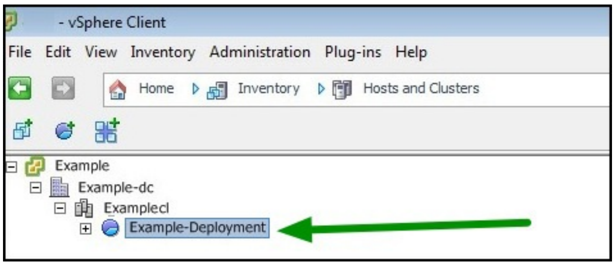
Note: The CID shown in Ops Manager is the name of the machine in vCenter.

Identifying PAS Jobs Using vCenter

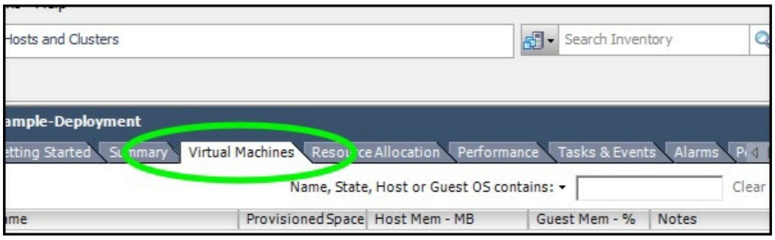
1. Launch the vSphere client and log in to the vCenter Server system.
2. Select the **Inventory > Hosts and Clusters** view.



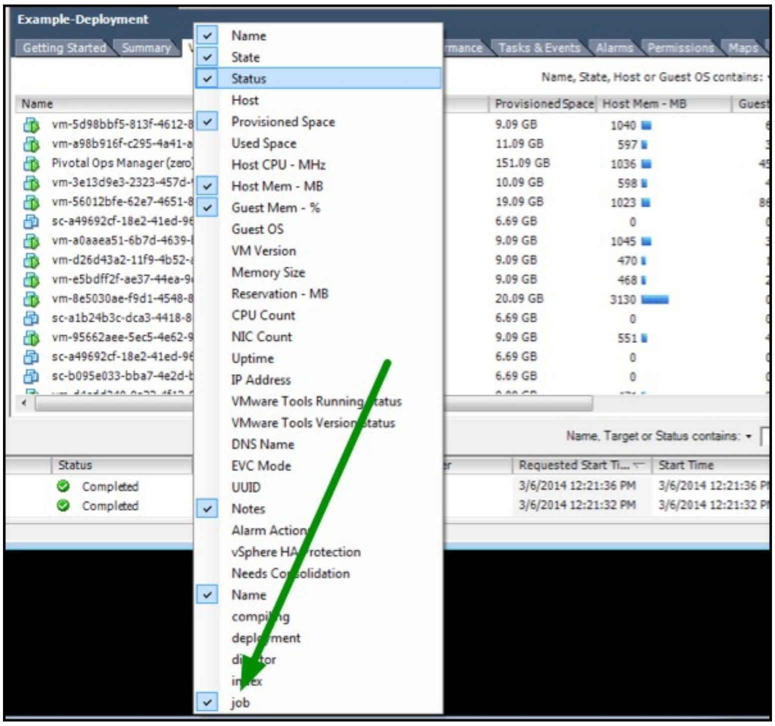
3. Select the Resource Pool containing your PAS deployment.



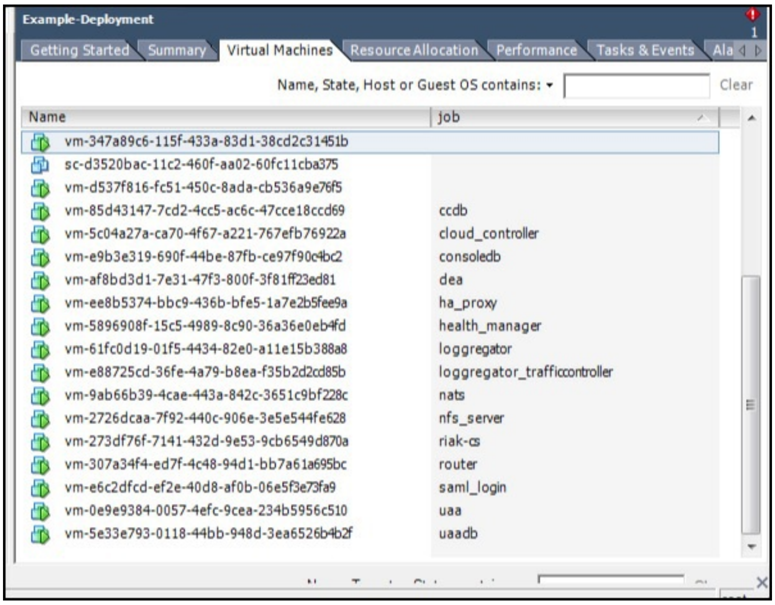
4. Select the **Virtual Machines** tab.



5. Right-click the column label heading and check **job**.



6. The job column displays the PAS job associated with each virtual machine.



Configuring Logging in PAS

Page last updated:

This topic describes the types of logs that Pivotal Cloud Foundry (PCF) generates. It also explains how to forward system logs to an external aggregator service, how to scale Loggregator component VMs to keep up with app log volume, and how to manage app traffic logging.

System Logs, App Logs, App Traffic Logs

Pivotal Cloud Foundry (PCF) generates two types of logs, *system logs* from PCF components and *app logs* from hosted apps, as differentiated in the table below:

Log Type	Originate from	Follow format	Stream from	Can to stream out to (configurable)	Visible to
System Logs	Platform components	Syslog standard	rsyslog agent	Component syslog drain	Operators
App Logs	Hosted apps	Format is up to the developer	Firehose[1]	External data platform, optionally via nozzles	Developers and Operators
		(Optional) With Syslog Adapter			
		Converted to syslog standard	Syslog Adapter	External syslog drain	

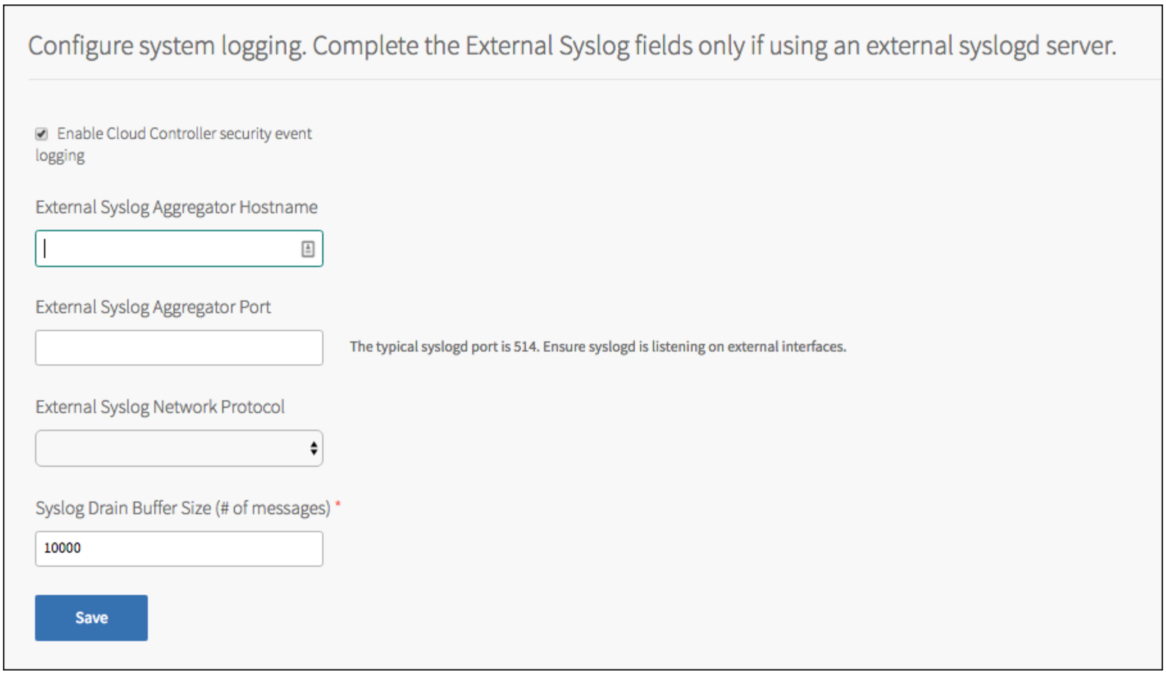
[1]The Loggregator Firehose also streams component metrics.

App traffic logs are system logs. When app containers communicate, or attempt to communicate, their host cells generate *app traffic logs*. App traffic logs are system logs, not app logs. These logs come from host cells, not apps, and they carry no information from within the app. App traffic logs only show app communication behavior, as detected from outside by the host cell.

Enable Syslog Forwarding

To forward system log messages to an external Reliable Event Logging Protocol (RELP) server, complete the following steps:

1. From the Pivotal Application Service (PAS) tile, Select **System Logging**.



2. To include security events in your log stream, select the **Enable Cloud Controller security event logging** checkbox. This logs all API requests, including the endpoint, user, source IP, and request result, in the Common Event Format (CEF).

3. Enter the IP address of your syslog server in **External Syslog Aggregator Hostname** and its port in **External Syslog Aggregator Port**. The default

port for a syslog server is `514`.

> **Note**: The host must be reachable from the PAS network, accept TCP connections, and use the RELP protocol. Ensure your syslog server listens on external interfaces.

4. Select an **External Syslog Network Protocol** to use when forwarding logs.

5. For the **Syslog Drain Buffer Size**, enter the number of messages the Doppler server can hold from Metron agents before the server starts to drop them. See the Loggregator Guide for Cloud Foundry Operators topic for more details.

6. Click **Save**.

Scale Loggregator

Apps constantly generate app logs and PCF platform components constantly generate component metrics. The Loggregator system combines these data streams and handles them as follows. See Overview of the Loggregator System for more information.

- The Metron agent running on each component or application VM collects and sends this data out to Doppler components.
- Doppler components temporarily buffer the data before periodically forwarding it to the Traffic Controller. When the log and metrics data input to a Doppler exceeds its buffer size for a given interval, data can be lost.
- The Traffic Controller serves the aggregated data stream through the Firehose WebSocket endpoint.

Follow the instructions below to scale the Loggregator system. For guidance on monitoring and capacity planning, see Monitoring Pivotal Cloud Foundry.

Add Component VM Instances

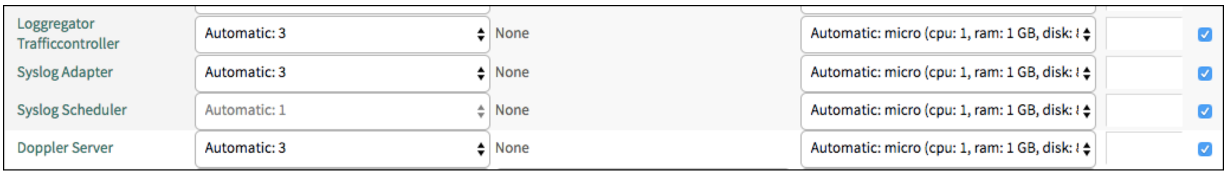
1. From the PAS tile, select **Resource Config**.

2. Increase the number in the **Instances** column of the component you want to scale. You can add instances for the following Loggregator components:
   - **Loggregator Traffic Controller**

     > **Note**: The Reverse Log Proxy (RLP) BOSH job is colocated on the Traffic Controller VM. If you want to scale Loggregator to handle more logs for syslog drains, you can add instances of the Traffic Controller.

     > **Note**: The BOSH System Metrics Forwarder job is colocated on the Traffic Controller VM. If you want to scale Loggregator to handle more BOSH system metrics, you can add instances of the Traffic Controller.

   - **Syslog Adapter**
   - **Doppler Server**



3. Click **Save**.

4. Click **Apply Changes**.

App Traffic Logging

App traffic logging generates logs when app containers communicate with each other directly, or attempt to communicate, as allowed by container-to-container networking (C2C) policies and Application Security Groups (ASGs).

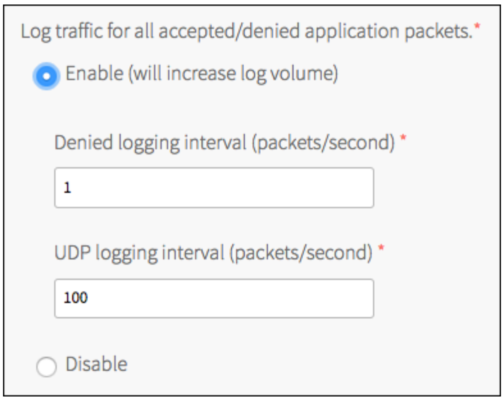
App traffic logging lets network security teams audit C2C traffic, by seeing allowed and denied packets, without needing access to the Cloud Controller or the apps themselves.

Enable App Traffic Logging

To enable app traffic logging:

1. From Ops Manager, navigate to the **Pivotal Application Service** tile > **Networking** pane.

2. Under **Log traffic for all accepted/denied application packets**, select **Enable (will increase log volume)** or **Disable** to enable or disable app traffic logging.



App Logging Behavior

App traffic logging generates log messages as follows:

- **TCP traffic** - Logs the first packet of every new TCP connection.
- **UDP traffic** - Logs UDP packets sent and received, up to a maximum per-second rate for each container. Set this rate limit in the **UDP logging interval** field (default: 100).
- **Packets denied** - Logs packets blocked by either a container-specific networking policy or by Application Security Group (ASG) rules applied across the space, org, or deployment. Logs packet denials up to a maximum per-second rate for each container, set in the **Denied logging interval** field (default: 1).

App Traffic Log Format and Contents

App traffic logs are formatted as described in the cf-networking-release Traffic logging documentation, following the `iptables-logger` format but without line breaks. For example, the first part of an app traffic log line looks like:


```
{"timestamp": "1500924070.182554722", "source": "cfnetworking.iptables", "message": "cfnetworking.iptables.ingress-allowed", "log_level": 1, "data": { "destination": { "container_id": "d5978989-1401-49ff-46cd-33e5","app_guid": "bc6f229d-5e4a-4c41-a63f-e8795496c283",
```


.

Each log message includes the following:

- Timestamp
- The GUID for the source or destination app that sent or was designated to receive the packet
- The protocol of the communication, `TCP` or `UDP`
- GUIDs for the container, space, and org running the source or destination app
- IP addresses and ports for both source and destination apps
- A `message` field recording whether the packet was allowed or denied, with one of the following four possibilities:
  - ASG allowed packet to exit source app container
  - C2C policy allowed packet to enter destination app container
  - ASG prevented packet from exiting source app container
  - C2C policy prevented packet from entering destination app container
- Additional information described in the cf-networking-release.

Denied Packet Causes

You can determine whether a denied-packet log resulted from a container networking policy or an ASG rule as follows:

- **Container networking policy**: Log `message` string includes `ingress-denied` and `packet direction` is `ingress`.
- **ASG rule**: Log `message` string includes `egress-denied` and `packet direction` is `egress`.

Global vs. ASG-Level App Traffic Logging

PCF supports two mechanisms for enabling app traffic logging. Setting **Log traffic** to **Enable** in Ops Manager enables app traffic logging globally for all ASGs and container policies. Setting the `log` property of an ASG to `true` enables app traffic logging at the individual ASG level.

Because these two mechanisms operate independently, PCF generates duplicate logs when app traffic logging is enabled globally and an ASG's `log` property is set to `true`. To avoid duplicate logs, Pivotal recommends setting the `log` property to `false` for all ASGs, or leaving it out entirely, when app traffic logging is enabled globally.

To focus on specific ASGs for log analysis, Pivotal recommends enabling app traffic logs globally and using a logging platform to filter traffic logs by ASG, rather than setting `log` at the individual ASG level.

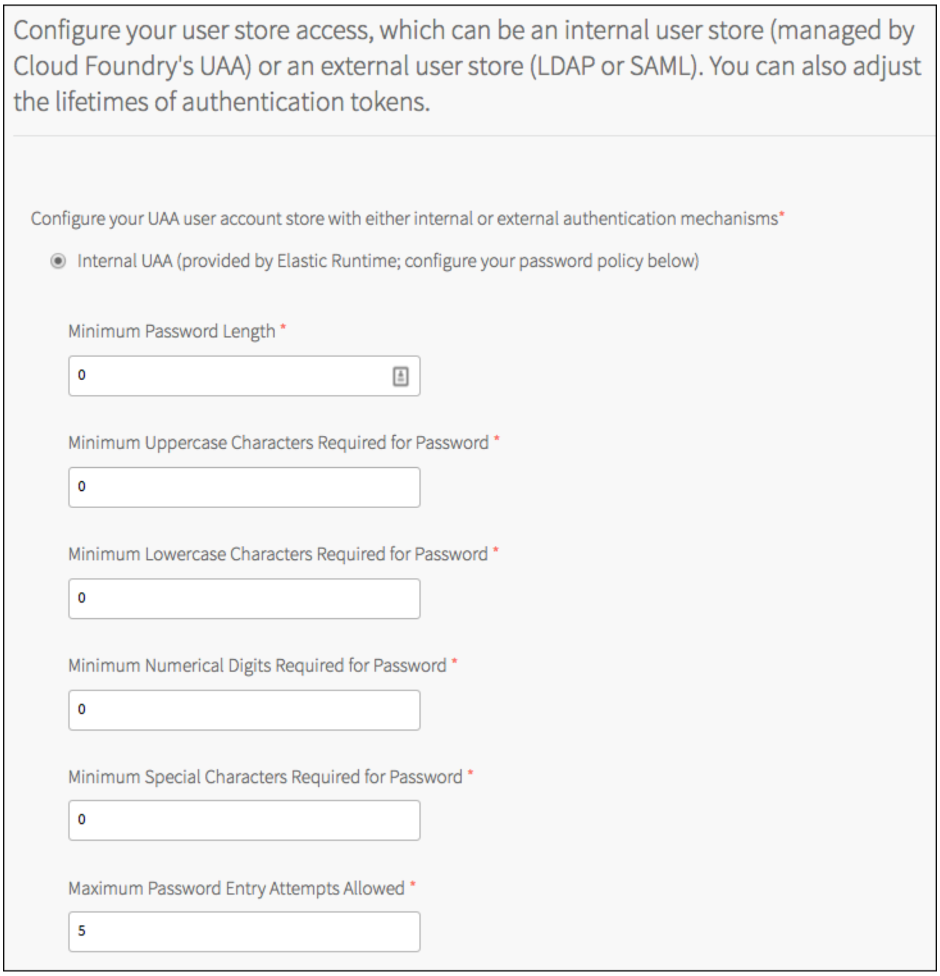
Configuring UAA Password Policy

Page last updated:

If your Pivotal Cloud Foundry (PCF) deployment uses the internal user store for authentication, you can configure its password policy within the Pivotal Application Service (PAS) tile.

Open the Internal UAA Configuration

1. In a browser, navigate to the fully qualified domain name (FQDN) of your Ops Manager and log in.
2. Click the **Pivotal Application Service** tile.
3. Select **Authentication and Enterprise SSO** on the **Settings** tab.



4. Confirm that the **Internal UAA** option is selected.

Set Password Requirements and Entry Attempts

1. For **Minimum Password Length**, enter the minimum number of characters for a valid password.
2. For **Minimum Uppercase Characters Required for Password**, enter the minimum number of uppercase characters required for a valid password.
3. For **Minimum Lowercase Characters Required for Password**, enter the minimum number of lowercase characters required for a valid password.
4. For **Minimum Numerical Digits Required for Password**, enter the minimum number of digits required for a valid password.
5. For **Minimum Special Characters Required for Password**, enter the minimum number of special characters required for a valid password.

6. For **Maximum Password Entry Attempts Allowed**, enter the maximum number of failures allowed to enter a password within a five-minute timespan before the account is locked.

Configuring Authentication and Enterprise SSO for PAS

Page last updated:

This topic describes Pivotal Cloud Foundry (PCF) Pivotal Application Service (PAS) authentication and single sign-on configuration with Lightweight Directory Access Protocol (LDAP) and Security Assertion Markup Language (SAML).

Refer to the instructions below to configure your deployment with SAML or LDAP.

Connecting Pivotal Application Service (PAS) to either the LDAP or SAML external user store allows the User Account and Authentication (UAA) server to delegate authentication to existing enterprise user stores.

If your enterprise user store is exposed as a SAML or LDAP Identity Provider for single sign-on (SSO), you can configure SSO to allow users to access the Apps Manager and Cloud Foundry Command Line Interface (cf CLI) without creating a new account or, if using SAML, without re-entering credentials.

See the Adding Existing SAML or LDAP Users to a PCF Deploymenttopic for information about managing user identity and pre-provisioning user roles with SAML or LDAP in PCF.

This Knowledge Base article explains the process used by the UAA Server when it attempts to authenticate a user through LDAP.

Configure PAS to Use a SAML Identity Provider

In SAML terminology, the SAML protocol communicates user data between an identity provider (IdP) and a service provider (SP).

To connect PAS with SAML, you must perform the following tasks:

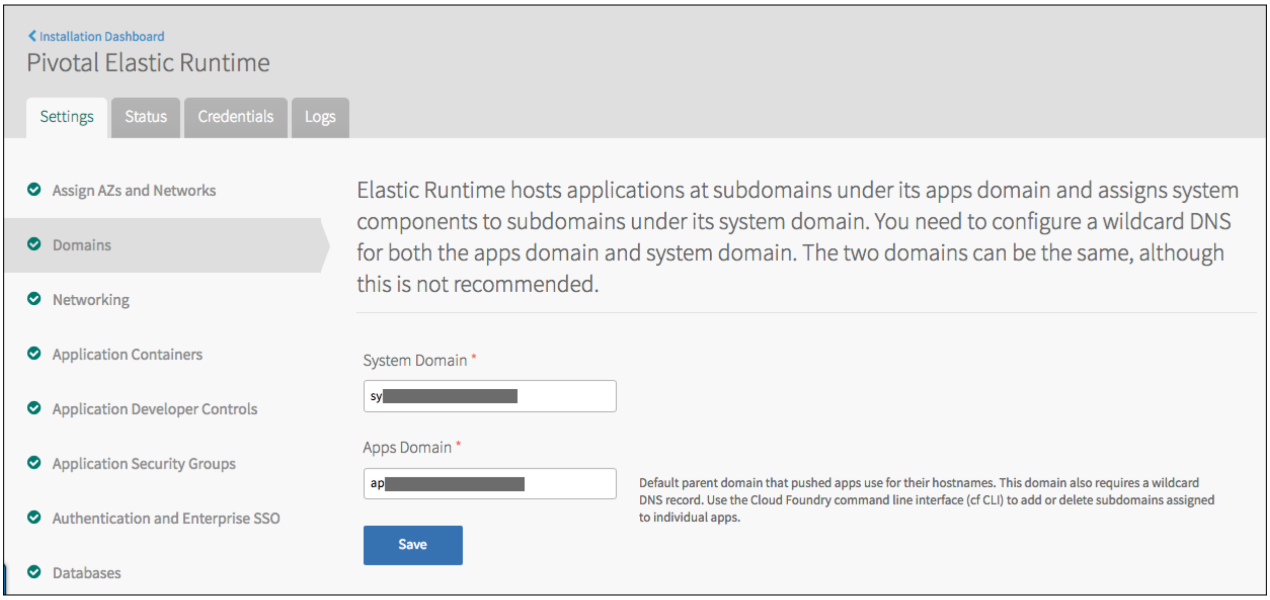
- Configure PAS as a Service Provider for SAML
- Configure SAML as an Identity Provider for PAS

Configure PAS as a Service Provider for SAML

The following procedure configures PAS to use a SAML IdP:

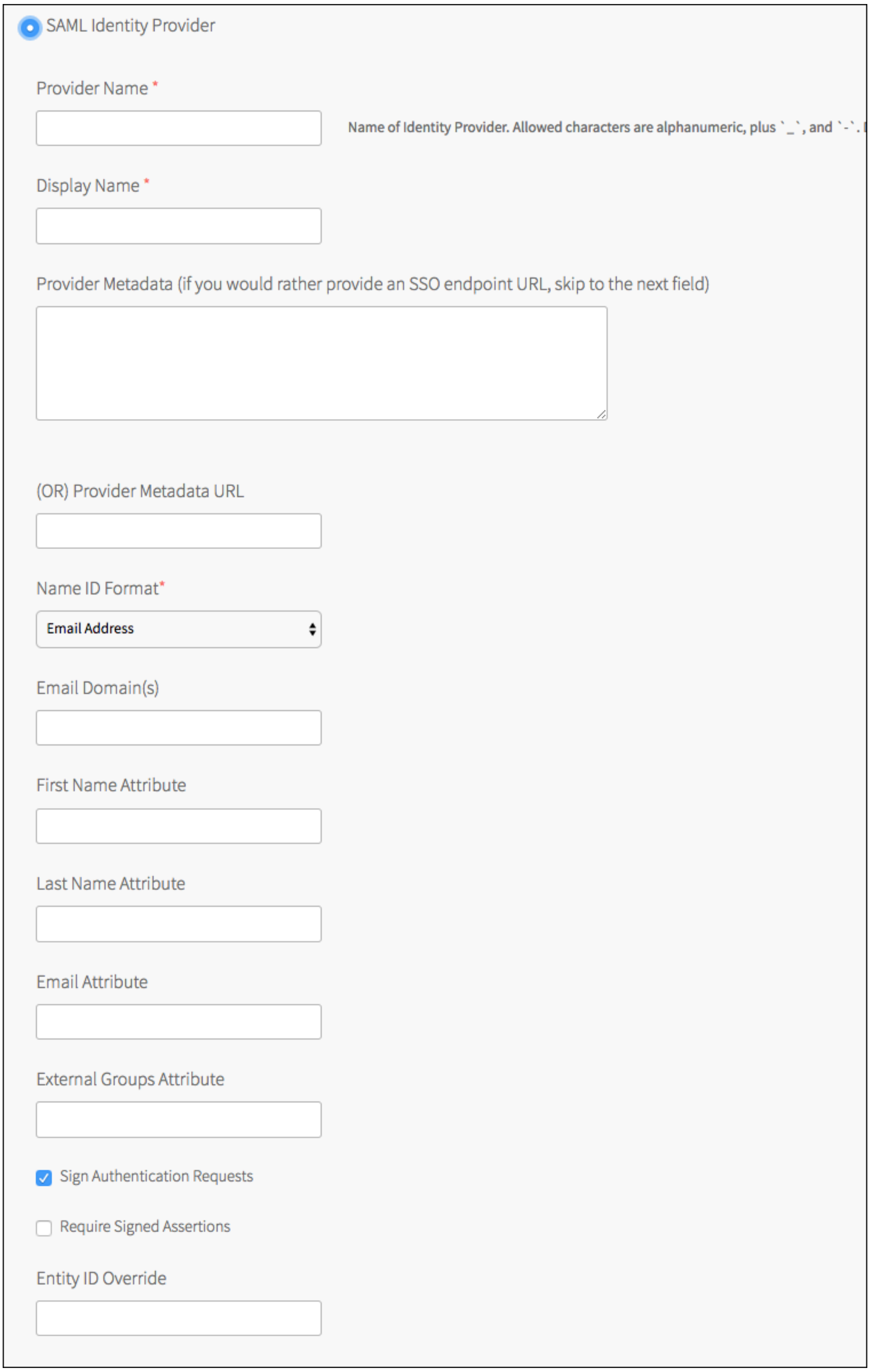
1. From the **Installation Dashboard**, click the PAS tile.

2. Select the **Domains** tab and record your system domain.



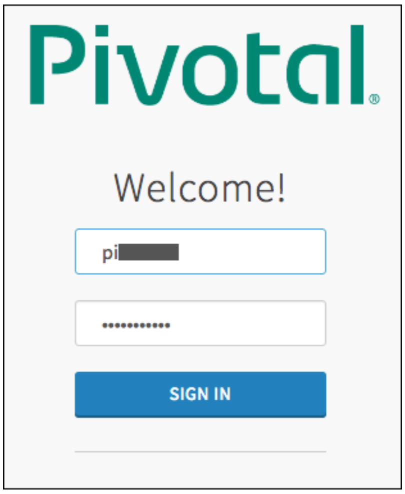
3. Select **Authentication and Enterprise SSO**.

4. Select **SAML Identity Provider**.



5. Set the **Provider Name**. This is a unique name you create for the Identity Provider. This name can include only alphanumeric characters, `+`, `_`, and `-`. You should not change this name after deployment because all external users use it to link to the provider.

6. Enter a **Display Name**. Your provider display name appears as a link on your Pivotal login page, which you can access at `https://login.YOUR-SYSTEM-DOMAIN`.



7. Retrieve the metadata from your Identity Provider and copy it into either the **Provider Metadata** or the **Provider Metadata URL** fields, depending on whether your Identity Provider exposes a Metadata URL. Refer to the [Configure SAML as an Identity Provider for PAS]() section of this topic for more information. Pivotal recommends that you use the Provider Metadata URL rather than Provider Metadata because the metadata can change. You can do this in either of the following ways:
   - If your Identity Provider exposes a Metadata URL, provide the Metadata URL.
   - Download your Identity Provider metadata and paste this XML into the **Provider Metadata** area.

   > **Note**: You only need to select one of the above configurations. If you configure both, your Identity Provider defaults to the **(OR) Provider Metadata URL**.

   > **Note**: Refer to the [Adding Existing SAML or LDAP Users to a PCF Deployment]() topic for information about on-boarding SAML users and mapping them to PAS user roles.

8. Select the **Name ID Format** for your SAML Identity Provider. This translates to `username` on PAS. The default is `Email Address`.

9. For **Email Domain(s)**, enter a comma-separated list of the email domains for external users who will receive invitations to Apps Manager.

10. For **First Name Attribute** and **Last Name Attribute**, enter the attribute names in your SAML database that correspond to the first and last names in each user record, for example `first_name` and `last_name`.

11. For **Email Attribute**, enter the attribute name in your SAML assertion that corresponds to the email address in each user record, for example `EmailID`.

12. For **External Groups Attribute**, enter the attribute name in your SAML database that defines the groups that a user belongs to, for example `group_memberships`. To map the groups from the SAML assertion to admin roles in PAS, follow the instructions in the [Grant Admin Permissions to an External Group (SAML or LDAP)]() section of the *Creating and Managing Users with the UAA CLI (UAAC)* topic.

13. By default, all SAML Authentication Request from PAS are signed. To change this, disable the **Sign Authentication Requests** checkbox and configure your Identity Provider to verify SAML authentication requests.

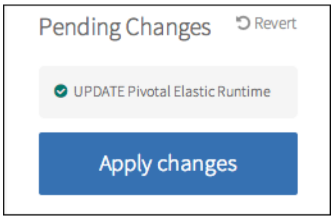
14. To validate the signature for the incoming SAML assertions, enable the **Required Signed Assertions** checkbox and configure your Identity Provider to send signed SAML assertions.

15. Your SAML Entity ID defaults to `http://login.YOUR-SYSTEM-DOMAIN` where `YOUR-SYSTEM-DOMAIN` is set in the **Domains** > **System Domain** field. To override this default, enter a custom SAML Entity ID in the **Entity ID Override** field.

16. Click **Save**.

17. Return to the **Installation Dashboard** by clicking the link.

18. On the Installation Dashboard, click **Apply Changes**.



Configure SAML as an Identity Provider for PAS

The following procedure configures a SAML IdP to designate PAS as an SP.

Download the Service Provider Metadata from `https://login.YOUR-SYSTEM-DOMAIN/saml/metadata`. Consult the documentation from your Identity Provider for configuration instructions.

Refer to the table below for information about certain industry-standard Identity Providers and how to integrate them with PAS:

Solution Name	Integration Guide
CA Single Sign-On aka CA SiteMinder	PDF
Ping Federate	PDF
Active Directory Federation Services	PDF

Note: Some Identity Providers allow uploads of Service Provider Metadata. Other providers require you to manually enter the Service Provider Metadata into a form.

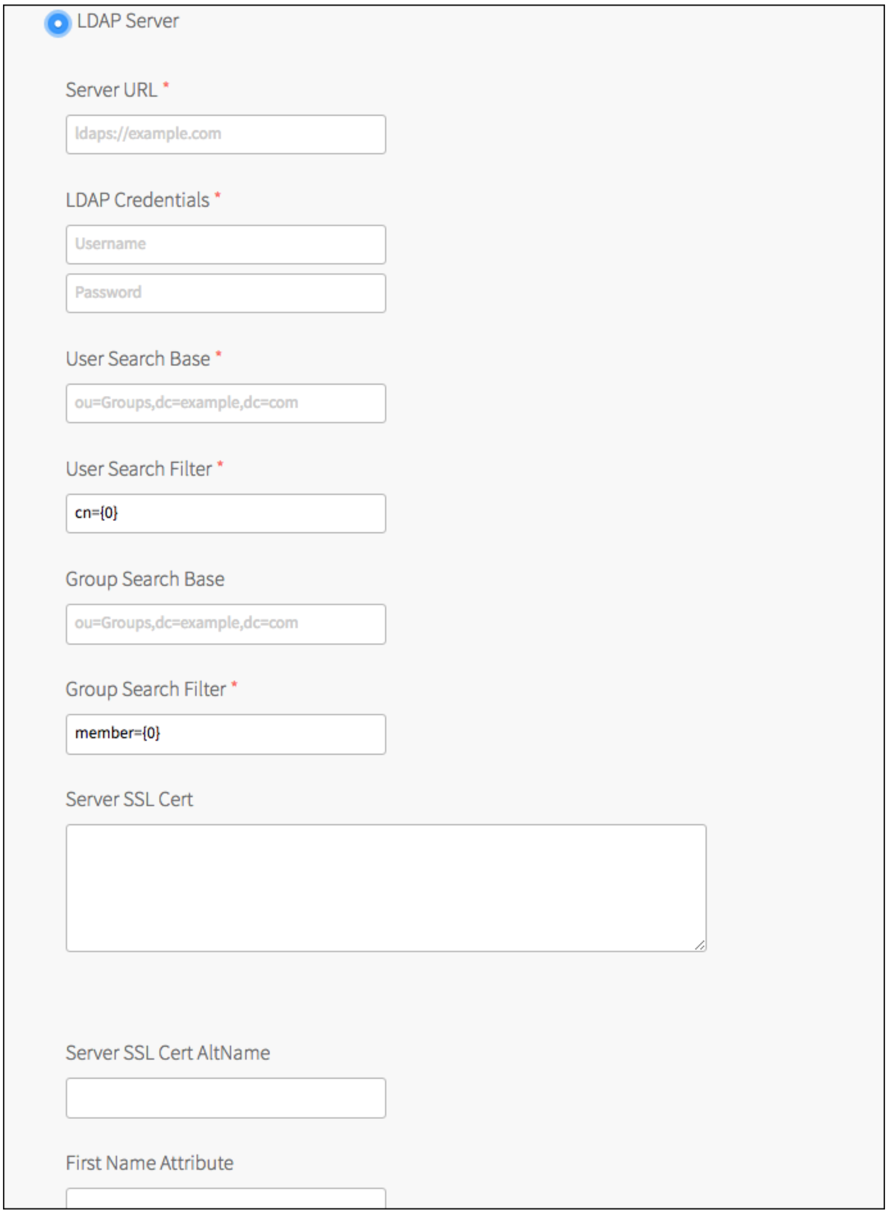
Configure LDAP as an Identity Provider for PAS

To integrate the UAA with one or more LDAP servers, configure PAS with your LDAP endpoint information as follows:

1. Log into the Operations Manager web interface.

2. On the Product Dashboard, select the PAS tile.

3. In the left navigation menu, select **Authentication and Enterprise SSO**.



4. Under **Configure your UAA**, select **LDAP Server**.

5. For **Server URL**, enter the URL(s) that point your LDAP server(s). With multiple LDAP servers, separate their URLs with spaces. Each URL must include one of the following protocols:
   - `ldap://` : This specifies that the LDAP server uses an unencrypted connection.
   - `ldaps://` : This specifies that the LDAP server uses SSL for an encrypted connection and requires that the LDAP server holds a trusted certificate or that you import a trusted certificate to the JVM truststore.

6. For **LDAP Credentials**, enter the LDAP Distinguished Name (DN) and password for binding to the LDAP Server. Example DN: `cn=administrator,ou=Users,dc=example,dc=com`

   Note: Pivotal recommends that you provide LDAP credentials that grant read-only permissions on the LDAP Search Base and the LDAP Group Search Base. In addition to this, if the bind user belongs to a different search base, you must use the full DN.

7. For **User Search Base**, enter the location in the LDAP directory tree from which any LDAP User search begins. The typical LDAP Search Base matches your domain name.

   For example, a domain named “cloud.example.com” typically uses the following LDAP User Search Base: `ou=Users,dc=example,dc=com`

8. For **User Search Filter**, enter a string that defines LDAP User search criteria. These search criteria allow LDAP to perform more effective and efficient searches. For example, the standard LDAP search filter `cn=Smith` returns all objects with a common name equal to `Smith`.

   In the LDAP search filter string that you use to configure PAS, use `{0}` instead of the username. For example, use `cn={0}` to return all LDAP objects with the same common name as the username.

In addition to `cn`, other attributes commonly searched for and returned are `mail`, `uid` and, in the case of Active Directory, `sAMAccountName`.

> **Note**: This Knowledge Base article provides instructions for testing and troubleshooting your LDAP search filters.

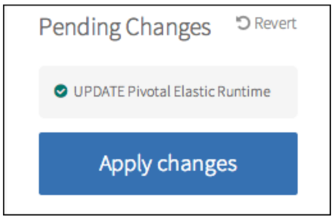
9. For **Group Search Base**, enter the location in the LDAP directory tree from which the LDAP Group search begins.

   For example, a domain named “cloud.example.com” typically uses the following LDAP Group Search Base: `ou=Groups,dc=example,dc=com`

   Follow the instructions in the Grant Admin Permissions to an External Group (SAML or LDAP) section of the *Creating and Managing Users with the UAA CLI (UAAC)* topic to map the groups under this search base to admin roles in PAS.

   > **Note**: Refer to the Adding Existing SAML or LDAP Users to a PCF Deployment topic to on-board individual LDAP users and map them to PAS Roles.

10. For **Group Search Filter**, enter a string that defines LDAP Group search criteria. The standard value is `member={0}`.
11. For **Server SSL Cert**, paste in the root certificate from your CA certificate or your self-signed certificate.
12. If you are using `ldaps://` with a self-signed certificate, enter a Subject Alternative Name for your certificate under **Server SSL Cert AltName**. Otherwise, leave this field blank.
13. For **First Name Attribute** and **Last Name Attribute**, enter the attribute names in your LDAP directory that correspond to the first and last names in each user record, for example `cn` and `sn`.
14. For **Email Attribute**, enter the attribute name in your LDAP directory that corresponds to the email address in each user record, for example `mail`.
15. For **Email Domain(s)**, enter a comma-separated list of the email domains for external users who will receive invitations to Apps Manager.
16. For **LDAP Referrals**, choose how the UAA handles LDAP server referrals out to other external user stores. The UAA can follow the external referrals, ignore them without returning errors, or throw an error for each external referral and abort the authentication.
17. Click **Save**.
18. Return to the **Installation Dashboard** by clicking the link.
19. On the Installation Dashboard, click **Apply Changes**.



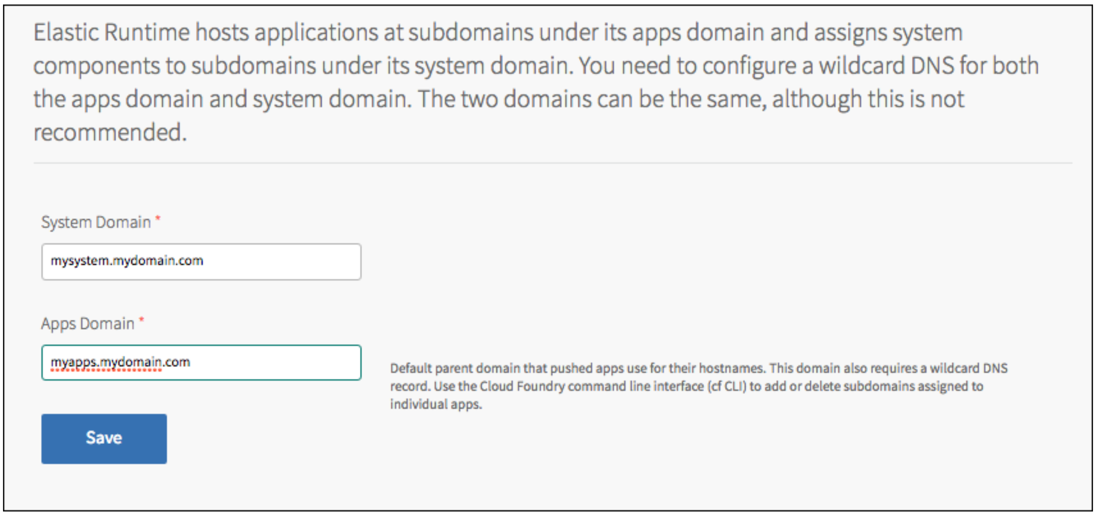
Switching Application Domains

Page last updated:

This topic describes how to change the domain of an existing Pivotal Cloud Foundry (PCF) installation, using an example domain change from `myapps.mydomain.com` to `newapps.mydomain.com`.

1. In PCF Ops Manager, select the **Pivotal Application Service** tile.

2. Select **Domains** from the menu to see the current **Apps Domain** for your Pivotal Application Service (PAS) deployment. In the following example it is `myapps.mydomain.com`.



3. In the terminal, run `cf login -a YOUR_API_ENDPOINT`. The cf CLI prompts you for your PCF username and password, as well as the org and space you want to access. See Identifying the API Endpoint for your PAS Instance if you don't know your API endpoint.

4. Run `cf domains` to view the domains in the space. If you have more than one shared domain, ensure that the domain you want to change is at the top of the list before you apply the new domain to your PAS tile configuration. You can delete and re-create the other shared domains as necessary to push the domain you want to change to the top of the list. If you do this, make sure to re-map the routes for each domain.

```
$ cf domains
Getting domains in org my-org as admin...

name                status
myapps.mydomain.com   shared
```

5. Run `cf routes` to confirm that your apps are assigned to the domain you plan to change.

```
$ cf routes
Getting routes as admin ...

space     host    domain              apps
my-space   myapp   myapps.mydomain.com   myapp
```

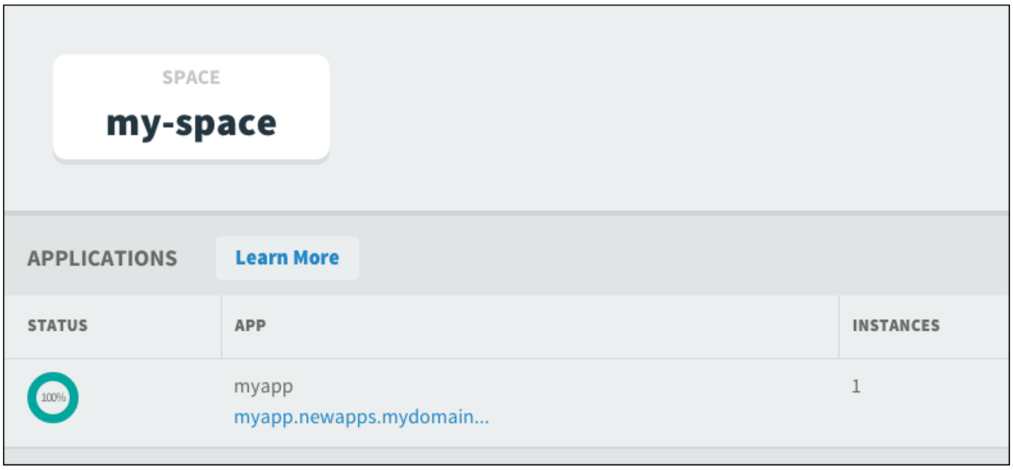
6. Run `cf create-shared-domain YOUR_DESIRED_NEW_DOMAIN` to create the new domain you want to use:

```
$ cf create-shared-domain newapps.mydomain.com

Creating shared domain newapps.mydomain.com as admin...
OK
```

7. Run `cf map-route APP_NAME NEW_DOMAIN -n HOST_NAME` to map the new domain to your app. In this example both the `NEW_DOMAIN` and `HOST_NAME` arguments are `myapp`, since this is both the name of the app to which we are mapping a route, and the intended hostname for the URL.

```
$ cf map-route myapp newapps.mydomain.com -n myapp

Creating route myapp.newapps.mydomain.com for org my-org / space my-space as admin...
OK
Adding route myapp.newapps.mydomain.com to app myapp in org my-org / space my-space as admin...
OK
```

8. Repeat the previous step for each app in this space. Afterwards, check Apps Manager to confirm that the route URL has updated correctly for each app:



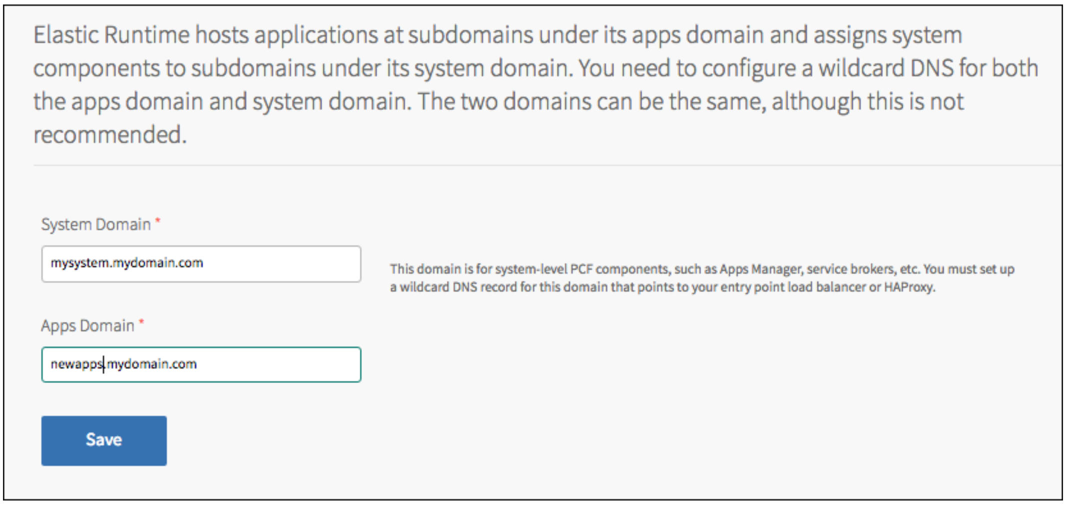
9. Repeat the above steps for each space in your PCF installation except for the System org, beginning with logging into the org and space and ending with confirming the URL update.

> **Note**: Ordinarily the System org contains only PCF apps that perform utility functions for your installation. Pivotal does not recommend pushing apps to this org. However, if you have pushed apps to System, you must also repeat the above steps for these apps.

10. Once you have confirmed that every app in every space has been mapped to the new domain, delete the old domain by running `cf delete-shared-domain OLD_DOMAIN_TO_DELETE`:

```
$ cf delete-shared-domain myapps.mydomain.com
Deleting domain myapps.mydomain.com as admin...

This domain is shared across all orgs.
Deleting it will remove all associated routes, and will make any app with this domain unreachable.
Are you sure you want to delete the domain myapps.mydomain.com?
> yes
OK
```

11. Configure your PAS tile to use the new domain, and apply changes. Apps that you push after your update finishes use this new domain.



Scaling PAS

Page last updated:

This topic discusses how to scale Pivotal Application Service (PAS) for different deployment scenarios. To increase the capacity and availability of the Pivotal Cloud Foundry (PCF) platform, and to decrease the chances of downtime, you can scale a deployment up using the instructions below.

If you want to make a PCF configuration highly available, see the High Availability in CF topic.

Note: In PCF 1.11 and later, PAS defaults to a highly available resource configuration.

Scaling Recommendations

The following table provides recommended instance counts for a high-availability deployment and the minimum instances for a functional deployment:

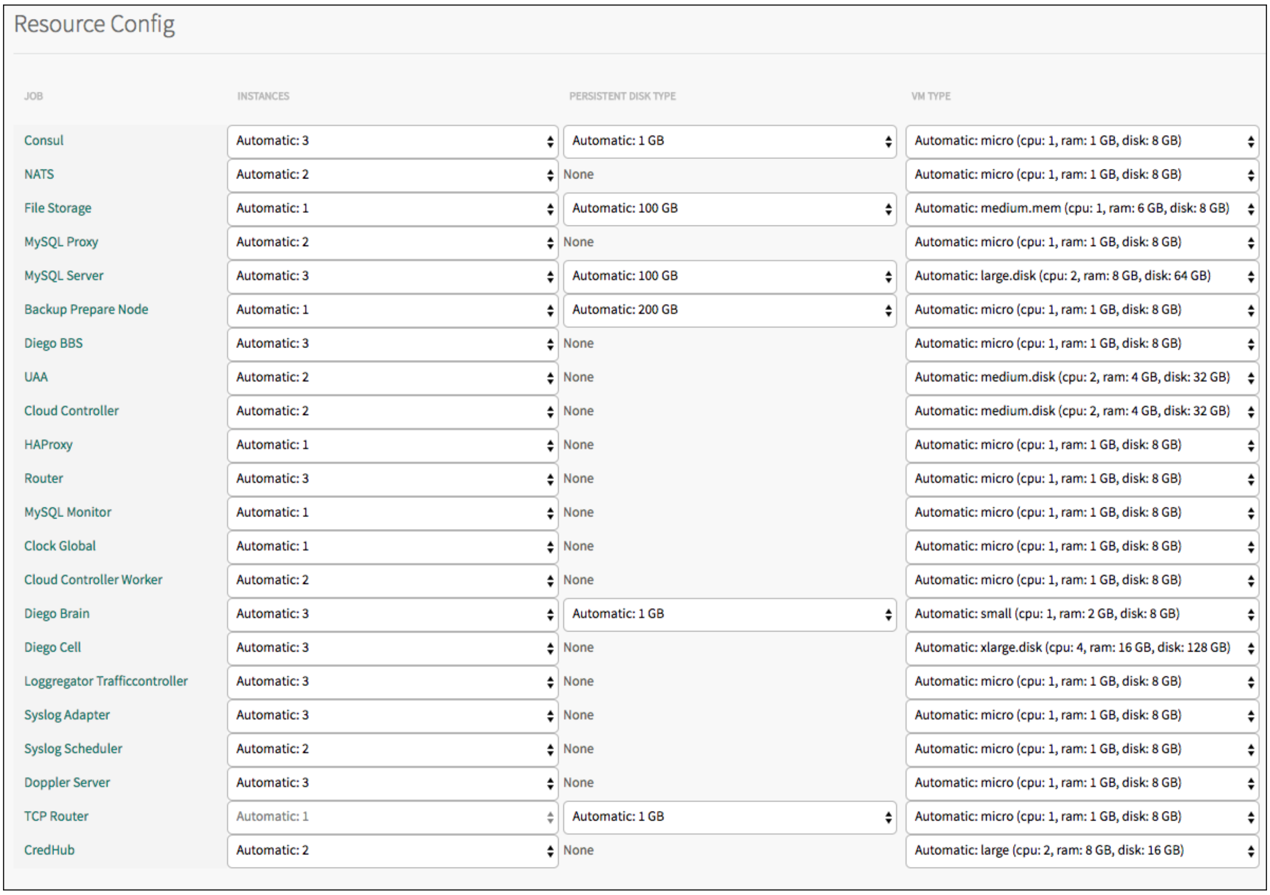
Pivotal Application Service (PAS) Job	Recommended Instance Number for HA	Minimum Instance Number	Notes
Diego Cell	≥ 3	1	The optimal balance between CPU/memory sizing and instance count depends on the performance characteristics of the apps that run on Diego cells. Scaling vertically with larger Diego cells makes for larger points of failure, and more apps go down when a cell fails. On the other hand, scaling horizontally decreases the speed at which the system rebalances apps. Rebalancing 100 cells takes longer and demands more processing overhead than rebalancing 20 cells.
Diego Brain	≥ 2	1	For high availability, use at least one per AZ, or at least two if only one AZ.
Diego BBS	≥ 3	1	Set this to an odd number equal to or one greater than the number of AZs you have, in order to maintain quorum. Distribute the instances evenly across the AZs, at least one instance per AZ.
Consul	≥ 3	1	Set this to an odd number equal to or one greater than the number of AZs you have, in order to maintain quorum. Distribute the instances evenly across the AZs, at least one instance per AZ.
MySQL Server	3	1	If you use an external database in your deployment, then you can set the MySQL Server instance count to `0`. For instructions about scaling down an internal MySQL cluster, see Scaling Down Your MySQL Cluster.
MySQL Proxy	2	1	If you use an external database in your deployment, then you can set the MySQL Proxy instance count to `0`.
NATS Server	≥ 2	1	In a high availability deployment, you might run a single NATS instance if your deployment lacks the resources to deploy two stable NATS servers. Components using NATS are resilient to message failures and the BOSH resurrector recovers the NATS VM quickly if it becomes non-responsive.
Cloud Controller	≥ 2	1	Scale the Cloud Controller to accommodate the number of requests to the API and the number of apps in the system.
Clock Global	≥ 2	1	For a high availability deployment, scale the Clock Global job to a value greater than 1 or to the number of AZs you have.
Router	≥ 2	1	Scale the router to accommodate the number of incoming requests. Additional instances increase available bandwidth. In general, this load is much less than the load on Diego cells.
HAProxy	0 or ≥ 2	0 or 1	For environments that require high availability, you can scale HAProxy to `0` and then configure a high-availability load balancer (LB) to point directly to each Gorouter instance. Alternately, you can also configure the high availability LB to point to HAProxy instance scaled at `≥ 2`. Either way, an LB is required to host Cloud Foundry domains at a single IP address.
UAA	≥ 2	1	
Doppler Server	≥ 2	1	Deploying additional Doppler servers splits traffic across them. For a high availability deployment, Pivotal recommends at least two per Availability Zone.
Loggregator Trafficcontroller	≥ 2	1	Deploying additional Loggregator Traffic Controllers allows you to direct traffic to them in a round-robin manner. For a high availability deployment, Pivotal recommends at least two per Availability Zone.
Syslog Scheduler	≥ 2	1	The Syslog Scheduler is a scalable component. For high availability, use at least one instance per AZ, or at least two instances if only one AZ is present.

> **Note:** You must configure a load balancer to achieve complete high-availability.

Scaling Up PAS

To scale up PAS instances, perform the following steps:

1. Navigate to the Pivotal Cloud Foundry Operations Manager Installation Dashboard.
2. Click the PAS tile in the Installation Dashboard.
3. Select **Resource Config**.



Resource Config

JOB	INSTANCES	PERSISTENT DISK TYPE	VM TYPE
Consul	Automatic: 3	Automatic: 1 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
NATS	Automatic: 2	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
File Storage	Automatic: 1	Automatic: 100 GB	Automatic: medium.mem (cpu: 1, ram: 6 GB, disk: 8 GB)
MySQL Proxy	Automatic: 2	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
MySQL Server	Automatic: 3	Automatic: 100 GB	Automatic: large.disk (cpu: 2, ram: 8 GB, disk: 64 GB)
Backup Prepare Node	Automatic: 1	Automatic: 200 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
Diego BBS	Automatic: 3	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
UAA	Automatic: 2	None	Automatic: medium.disk (cpu: 2, ram: 4 GB, disk: 32 GB)
Cloud Controller	Automatic: 2	None	Automatic: medium.disk (cpu: 2, ram: 4 GB, disk: 32 GB)
HAProxy	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
Router	Automatic: 3	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
MySQL Monitor	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
Clock Global	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
Cloud Controller Worker	Automatic: 2	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
Diego Brain	Automatic: 3	Automatic: 1 GB	Automatic: small (cpu: 1, ram: 2 GB, disk: 8 GB)
Diego Cell	Automatic: 3	None	Automatic: xlarge.disk (cpu: 4, ram: 16 GB, disk: 128 GB)
Loggregator Trafficcontroller	Automatic: 3	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
Syslog Adapter	Automatic: 3	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
Syslog Scheduler	Automatic: 2	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
Doppler Server	Automatic: 3	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
TCP Router	Automatic: 1	Automatic: 1 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
CredHub	Automatic: 2	None	Automatic: large (cpu: 2, ram: 8 GB, disk: 16 GB)

4. To scale your deployment horizontally, increase the number of **Instances** of a job. See Scaling Recommendations for guidance on the number of job instances required to ensure high availability.

   > **Note:** In PCF 1.12, you cannot scale the Autoscaler job to greater than one instance.

5. To scale your deployment vertically, adjust the **Persistent Disk Type** and **VM Type** of a job to allocate more disk space and memory. If you choose **Automatic** from the drop-down menu, PAS uses the recommended amount of resources for the job.
6. Click **Save**.
7. Return to the **Installation Dashboard** and click **Apply Changes**.

Scaling Down PAS

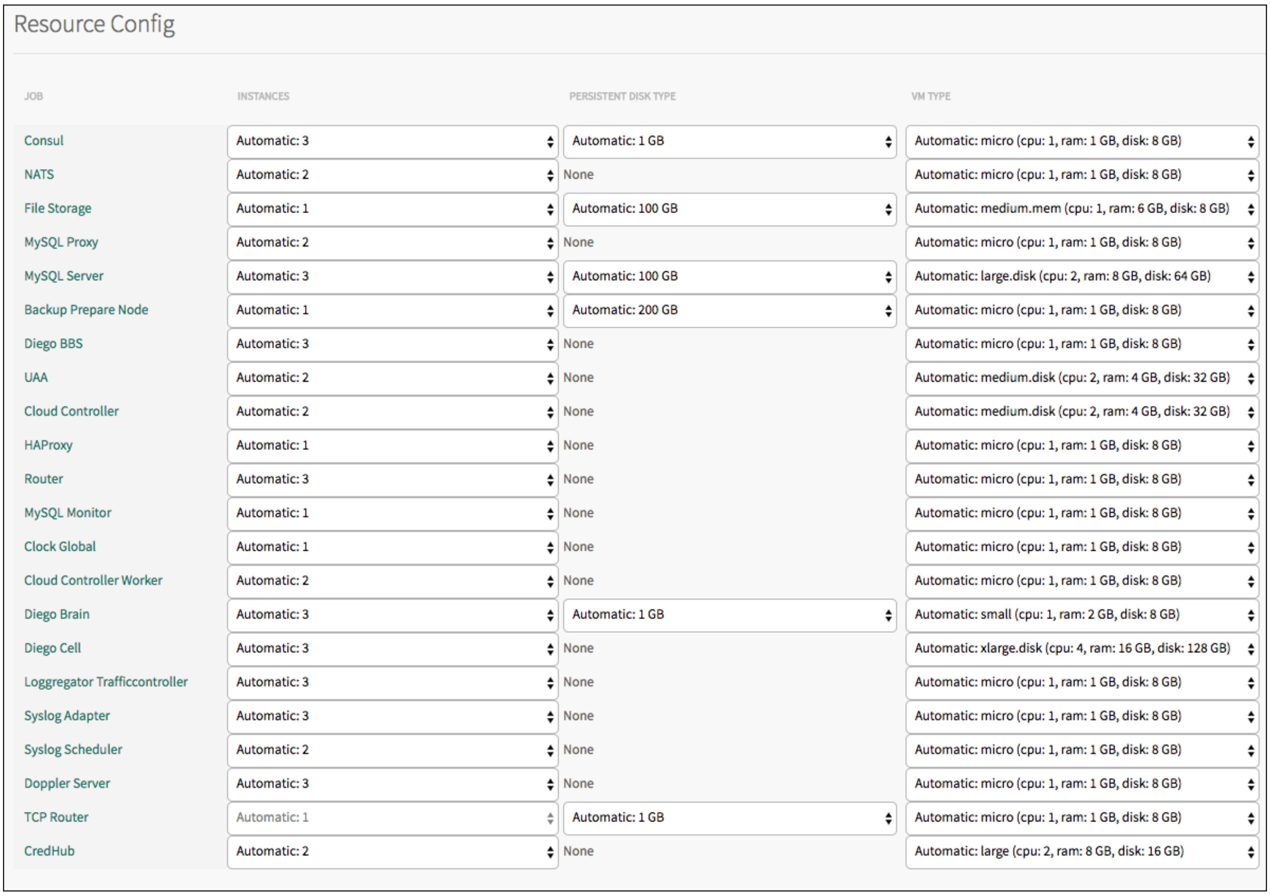
If you are deploying a PCF configuration that does not need to be highly available, Pivotal recommends scaling down job instances to the minimum number required for a functional deployment.

To scale down your deployment, perform the following steps:

1. Navigate to the Pivotal Cloud Foundry Operations Manager Installation Dashboard.

2. Click the PAS tile in the Installation Dashboard.

3. Select **Resource Config**.



Resource Config

JOB	INSTANCES	PERSISTENT DISK TYPE	VM TYPE
Consul	Automatic: 3	Automatic: 1 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
NATS	Automatic: 2	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
File Storage	Automatic: 1	Automatic: 100 GB	Automatic: medium.mem (cpu: 1, ram: 6 GB, disk: 8 GB)
MySQL Proxy	Automatic: 2	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
MySQL Server	Automatic: 3	Automatic: 100 GB	Automatic: large.disk (cpu: 2, ram: 8 GB, disk: 64 GB)
Backup Prepare Node	Automatic: 1	Automatic: 200 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
Diego BBS	Automatic: 3	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
UAA	Automatic: 2	None	Automatic: medium.disk (cpu: 2, ram: 4 GB, disk: 32 GB)
Cloud Controller	Automatic: 2	None	Automatic: medium.disk (cpu: 2, ram: 4 GB, disk: 32 GB)
HAProxy	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
Router	Automatic: 3	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
MySQL Monitor	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
Clock Global	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
Cloud Controller Worker	Automatic: 2	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
Diego Brain	Automatic: 3	Automatic: 1 GB	Automatic: small (cpu: 1, ram: 2 GB, disk: 8 GB)
Diego Cell	Automatic: 3	None	Automatic: xlarge.disk (cpu: 4, ram: 16 GB, disk: 128 GB)
Loggregator Trafficcontroller	Automatic: 3	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
Syslog Adapter	Automatic: 3	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
Syslog Scheduler	Automatic: 2	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
Doppler Server	Automatic: 3	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
TCP Router	Automatic: 1	Automatic: 1 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
CredHub	Automatic: 2	None	Automatic: large (cpu: 2, ram: 8 GB, disk: 16 GB)

4. In the **Resource Config** screen, decrease the number of **Instances** for each job. Choose the suggested values outlined in Scaling Recommendations, or in the Scaling Recommendations for Specific Deployment Configurations and click **Save**.

5. Return to the **Installation Dashboard** and click **Apply Changes**.

Scaling Down Jobs with Persistent Disk

If you scale down or delete a job that uses persistent disk, PAS marks the disk as “orphaned.” Orphaned disks are not attached to any job, and PAS deletes them after five days.

Use the BOSH CLI to list and recover orphaned disks. Follow the instructions in the “Prepare to Use the BOSH CLI” section of the “Advanced Troubleshooting with the BOSH CLI” topic to log in to the BOSH Director, and then follow the procedures in “Orphaned Disks” in the BOSH documentation.

Scaling Recommendations for Specific Deployment Configurations

If you are using one of the following configurations, choose the values in the corresponding table to scale instances for your particular deployment:

- Deployments Using External Databases
- Deployments Using Internal MySQL
- Deployments Using Internal MySQL Databases
- Deployments Using an External Blobstore
- Deployments Using External Load Balancers

Deployments Using External Databases

If you are using an external database, you can scale down the instance counts for internal MySQl jobs.

Select the following values in the Resource Config:

Job	Instance Count
MySQL Server	0
MySQL Proxy	0

Deployments Using Internal MySQL

If you are using the internal MySQL database on a clean install, or on an upgrade from a configuration that previously used internal MySQL databases, you do not need to change the default values shown in the table below.

If you need to change back to this configuration, choose the values shown below in the Resource Config.

Note: Changing back to this configuration deletes any data written to your other database option.

Job	Instance Count
MySQL Server	3
MySQL Proxy	2

Note: Apps that are not using MySQL for PCF are not affected by the scaling process when you redeploy PAS. In addition, redeploying PAS with the MySQL cluster means that the PCF API will not be available for a brief period of time. For example, you are not able to push apps or query their state during this time.

Deployments Using an External Blobstore

If you are using an external Blobstore, choose the following value in the Resource Config:

Job	Instance Count
File Storage	0

Deployments Using External Load Balancers

If you are using an external load balancer, choose the following values in the Resource Config:

Job	Instance Count
HAProxy	0
Router	≥ 1
Diego Brain	≥ 1

For more information about configuring load balancers in the Resource Config section of PAS, see the “Deploying PAS” topic for your IaaS. For example:

- Deploying PAS on AWS
- Deploying PAS on Azure
- Deploying PAS on GCP
- Installing PAS after Deploying Pivotal Cloud Foundry on OpenStack
- Configuring PAS for vSphere

Scaling Down Your MySQL Cluster

This topic describes how to safely scale down your MySQL cluster to a single node.

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

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

Check the Health of Your Cluster

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

1. Use the Cloud Foundry Command Line Interface (cf CLI) to target the API endpoint of your Pivotal Cloud Foundry (PCF) deployment:

```
$ cf api api.YOUR-SYSTEM-DOMAIN
Setting api endpoint to api.YOUR-SYSTEM-DOMAIN...
OK


API endpoint:   https://api.YOUR-SYSTEM-DOMAIN... (API version: 2.54.0)
Not logged in. Use 'cf login' to log in.
```

2. Log in with your User Account and Authentication (UAA) Administrator user credentials. Obtain these credentials by clicking the **Credentials** tab of the Pivotal Application Service (PAS) tile, locating the **Admin Credentials** entry in the **UAA** section, and clicking **Link to Credential**.

```
$ cf login -u admin
API endpoint: https://api.YOUR-SYSTEM-DOMAIN


Password>
Authenticating...
OK
```

3. Create a test organization to verify the database across all nodes:

```
$ cf create-org data-integrity-test-organization
Creating org data-integrity-test-organization as admin...
OK


Assigning role OrgManager to user admin in org data-integrity-test-organization ...
OK


TIP: Use 'cf target -o data-integrity-test-organization' to target new org
```

4. Obtain the IP addresses of your MySQL server by performing the following steps:
   a. From the PCF **Installation Dashboard**, click the **Pivotal Application Service** tile.
   b. Click the **Status** tab.
   c. Record the IP addresses for all instances of the **MySQL Server** job.

5. Retrieve Cloud Controller database credentials from CredHub using the Ops Manager API:
   a. Perform the procedures in the [Using the Ops Manager API] topic to authenticate and access the Ops Manager API.
   b. Use the `GET /api/v0/deployed/products` endpoint to retrieve a list of deployed products, replacing `UAA-ACCESS-TOKEN` with the access token recorded in the previous step:

```
$ curl "https://OPS-MAN-FQDN/api/v0/deployed/products" \
-X GET \
-H "Authorization: Bearer UAA-ACCESS-TOKEN"
```

   c. In the response to the above request, locate the product with an `installation_name` starting with `cf-` and copy its `guid`.
   d. Run the following `curl` command, replacing `PRODUCT-GUID` with the value of `guid` from the previous step:

```
$ curl "https://OPS-MAN-FQDN/api/v0/deployed/products/PRODUCT-GUID/variables?name=cc-db-credentials" \
-X GET \
-H "Authorization: Bearer UAA-ACCESS-TOKEN"
```

e. Record the Cloud Controller database `username` and `password` from the response to the above request.

6. SSH into the Ops Manager VM. Because the procedures vary by IaaS, review the SSH into Ops Manager section of the Advanced Troubleshooting with the BOSH CLI topic for specific instructions.

7. For each of the MySQL server IP addresses recorded above, perform the following steps from the Ops Manager VM:

   a. Query the new organization with the following command, replacing `YOUR-IP` with the IP address of the MySQL server and `YOUR-IDENTITY` with the `identity` value of the CCDB credentials obtained above:

   ```
   $ mysql -h YOUR-IP -u YOUR-IDENTITY -D ccdb -p -e "select created_at, name from organizations where name = 'data-integrity-test-organization'"
   ```

   b. When prompted, provide the `password` value of the CCDB credentials obtained above.
   c. Examine the output of the `mysql` command and verify the `created_at` date is recent.

   ```
   +---------------------+----------------------------------+
   | created_at          | name                             |
   +---------------------+----------------------------------+
   | 2016-05-28 01:11:42 | data-integrity-test-organization |
   +---------------------+----------------------------------+
   ```

8. If each MySQL server instance does not return the same `created_at` result, contact Pivotal Support before proceeding further or making any changes to your deployment. If each MySQL server instance does return the same result, then you can safely proceed to scaling down your cluster to a single node by performing the steps in the following section.

Scale Down Your Cluster

1. From the PCF **Installation Dashboard**, click the **Pivotal Application Service** tile.

2. Select **Resource Config**.

3. Use the drop-down menu to change the **Instances** count for **MySQL Server** to `1`.

4. Click **Save** to apply the changes.

5. Delete your test organization with the following cf CLI command:

```
$ cf delete-org data-integrity-test-organization
```

Using Docker Registries

Page last updated:

This topic describes how to configure Pivotal Cloud Foundry (PCF) to access Docker registries such as Docker Hub, by using either a root certificate authority (CA) certificate or by adding its IP address to a whitelist. It also explains how to configure PCF to access Docker registries through a proxy.

Docker registries store Docker container images. PCF uses these images to create the Docker containers that it runs apps in.

Prerequisite: Enable Docker Support

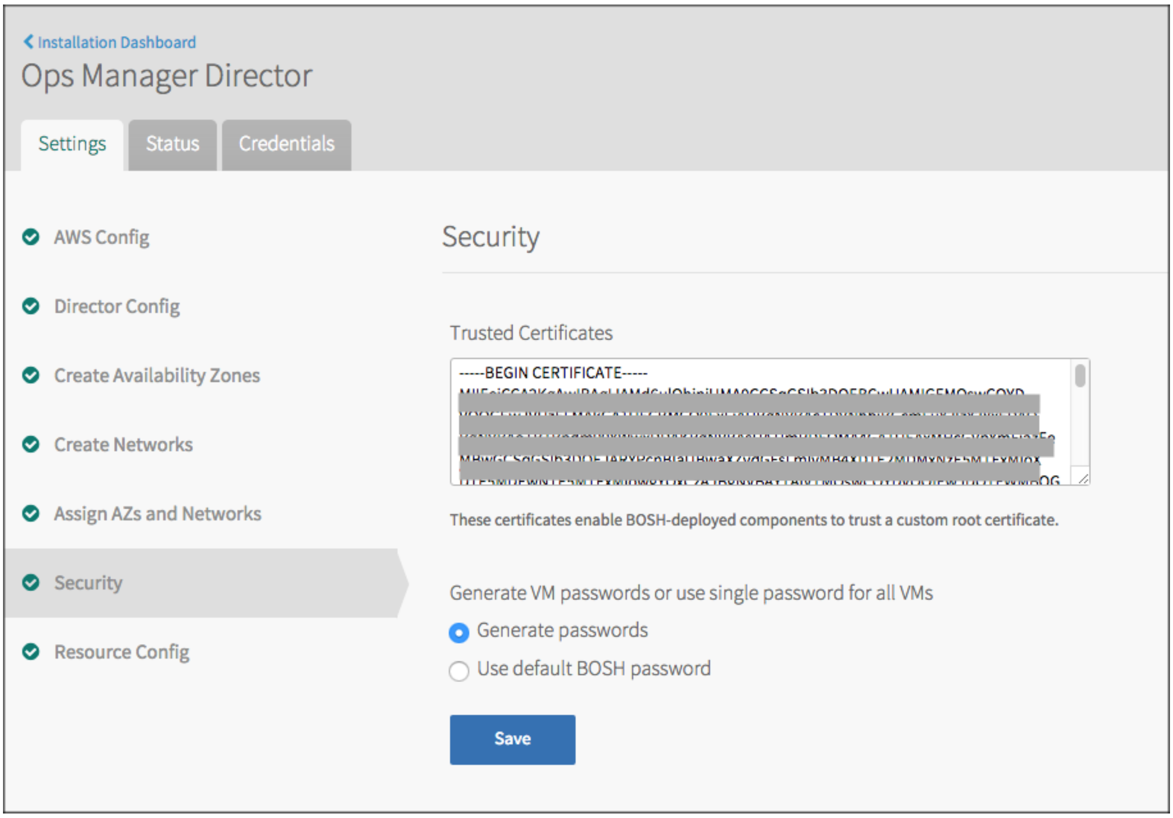
PCF can only access Docker registries if an operator has enabled Docker support with the `cf enable-feature-flag diego_docker` command, as described in the Using Docker in Cloud Foundry topic.

With Docker enabled, developers can push an app with a Docker image using the Cloud Foundry Command Line Interface (cf CLI). For more information, see the Deploy an App with Docker topic.

Use a CA Certificate

If you provide your root CA certificate in the Ops Manager configuration, follow this procedure:

1. In the Ops Manager Installation Dashboard, click the **Ops Manager Director** tile.
2. Click **Security**.



3. In the **Trusted Certificates** field, paste one or more root CA certificates. The Docker registry does not use the CA certificate itself but uses a certificate that is signed by the CA certificate.
4. Click **Save**.
5. Choose one of the following:
   - If you are configuring Ops Manager for the first time, return to your specific IaaS installation instructions (AWS, Azure, GCP, OpenStack, vSphere) to continue the installation process.

- If you are modifying an existing Ops Manager installation, return to the Ops Manager Installation Dashboard and click **Apply Changes**.

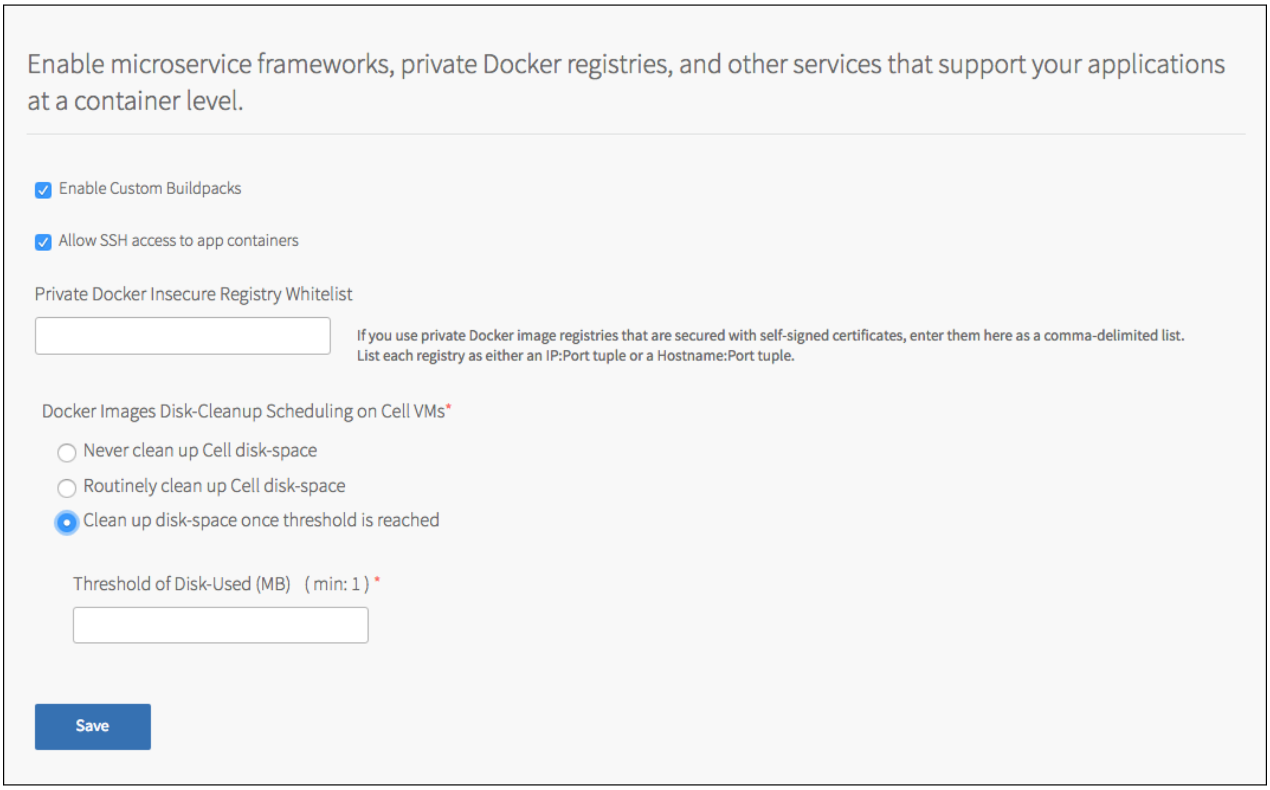
After configuration, BOSH propagates your CA certificate to all application containers in your deployment. You can then push and pull images from your Docker registries.

Use an IP Address Whitelist

If you choose not to provide a CA certificate, you must provide the IP address of your Docker registry.

> **Note**: Using a whitelist skips SSL validation. If you want to enforce SSL validation, enter the IP address of the Docker registry in the **No proxy** field described below.

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the **Pivotal Application Service** tile, and navigate to the **Application Containers** tab.



3. Select **Enable Custom Buildpacks** to enable custom-built application runtime buildpacks.
4. Select **Allow SSH access to app containers** to enable app containers to accept SSH connections. If you use a load balancer instead of HAProxy, you must open port 2222 on your load balancer to enable SSH traffic. To open an SSH connection to an app, a user must have Space Developer privileges for the space where the app is deployed. Operators can grant those privileges in Apps Manager or using the cf CLI.
5. For **Private Docker Insecure Registry Whitelist**, provide the hostname or IP address and port that point to your private Docker registry. For example, enter `198.51.100.1:80` or `mydockerregistry.com:80`. Enter multiple entries in a comma-delimited sequence. SSL validation is ignored for private Docker image registries secured with self-signed certificates at these locations.
6. Under **Docker Images Disk-Cleanup Scheduling on Cell VMs**, choose one of the options listed below. For more information about these options, see Configuring Docker Images Disk-Cleanup Scheduling.
   - **Never clean up Cell disk-space**
   - **Routinely clean up Cell disk-space**
   - **Clean up disk-space once threshold is reached**. If you choose this option, enter the amount of disk space limit the Cell must reach before disk cleanup initiates under **Threshold of Disk-Used (MB)**.
7. Click **Save**.
8. Choose one of the following:

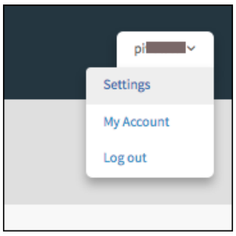
- If you are configuring Pivotal Application Service (PAS) for the first time, return to your specific IaaS installation instructions (AWS, Azure, GCP, OpenStack, vSphere) to continue the installation process.
- If you are modifying an existing PAS installation, return to the Ops Manager Installation Dashboard and click **Apply Changes**.

After configuration, PAS allows Docker images to pass through the specified IP address without checking certificates.

Set Proxies for Docker Registries

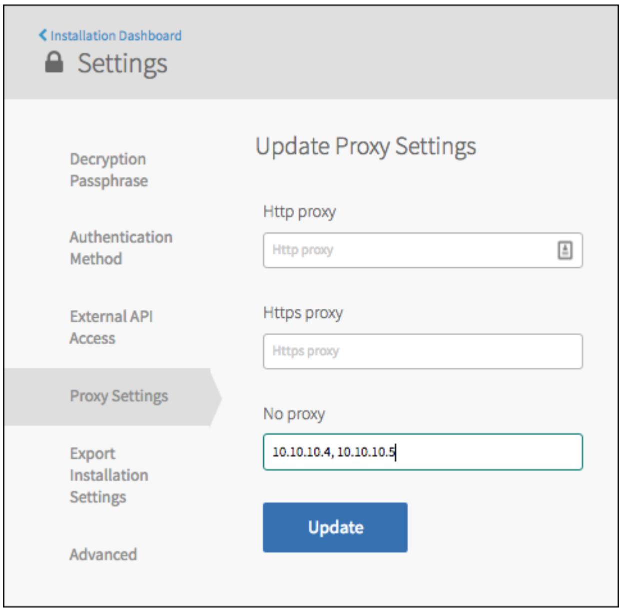
To configure PCF to access a Docker registry through a proxy, do the following:

1. On the Installation Dashboard, navigate to USERNAME > **Settings** > **Proxy Settings**.



2. On the **Update Proxy Settings** pane, complete one of the following fields:
   - **HTTP proxy**: If you have an HTTP proxy server for your Docker registry, enter its IP address.
   - **HTTPS proxy**: If you have an HTTPS proxy server for your Docker registry, enter its IP address.
   - **No proxy**: If you do not use a proxy server, enter the IP address for the Docker registry. This field may already contain proxy settings for the BOSH Director.

   Enter multiple IP addresses as a comma-separated list.



3. Click **Update**.

Configuring Cell Disk Cleanup Scheduling

Page last updated:

This topic describes how to configure disk cleanup scheduling on Diego cells in Pivotal Cloud Foundry (PCF).

What is Disk Cleanup

PCF isolates application instances (AIs) from each other using containers that run inside Diego cells. Containers enforce a set of isolation layers including file system isolation. A PCF container file system can either be a PCF stack or the result of pulling a Docker image.

For performance reasons, the cells cache the Docker image layers and the PCF stacks used by running AIs. When PCF destroys an AI or reschedules an AI to a different cell, a chance exist that certain Docker image layers or an old PCF stack becomes unused. If PCF does not clean these unused layers, the cell ephemeral disk will slowly fill.

Disk cleanup is the process of removing unused layers from the cell disk. The disk cleanup process removes all unused Docker image layers and old PCF Stacks, regardless of their size or age.

Options for Disk Cleanup

PCF provides the following options for scheduling the disk cleanup process on Diego cells:

- **Never clean up the Cell Disk**: Pivotal does not recommend using this option for production environments.
- **Routinely clean up the Cell Disk**: This option makes the cell schedule a disk cleanup whenever a container is created. Running the disk cleanup process this frequently can result in a negative impact on the cell performance.
- **Clean up Disk space once a threshold is reached**: This option makes the cell schedule the disk cleanup process only when a configurable disk space threshold is reached or exceeded.

See the Configure Disk Cleanup Scheduling section of this topic to select one of these options.

Recommendations

Selecting the best option for disk cleanup depends on the workload that the Diego cells run.

For PCF installations that primarily run buildpack-based apps, Pivotal recommends selecting the **Routinely clean up Cell Disk space** option. The **Routinely clean up Cell Disk space** option ensures that when a new stack becomes available on a cell, the old stack is dropped immediately from the cache.

For PCF installations that primarily run Docker images, or both Docker images and buildpack-based apps, Pivotal recommends selecting the **Clean up Disk space once a threshold is reached** option with a reasonable threshold.

Calculating a Threshold

To calculate a realistic value when configuring the disk cleanup threshold, you must identify some of the most frequently used Docker images in your PCF installation. Docker images tend to be constructed by creating layers over existing, base, images. In some cases, you may find it easier to identify which base Docker images are most frequently used.

Follow the steps below to calculate the disk cleanup threshold:

1. Identify the most frequently used Docker images or base Docker images.
2. Using the Docker CLI, measure the size of those images.

   For example, you may determine the most frequently used images in a test deployment are `openjdk:7`, `nginx:1.13`, and `php:7-apache`. In this case, you can run the following commands to pull the identified images locally, then measure their sizes:

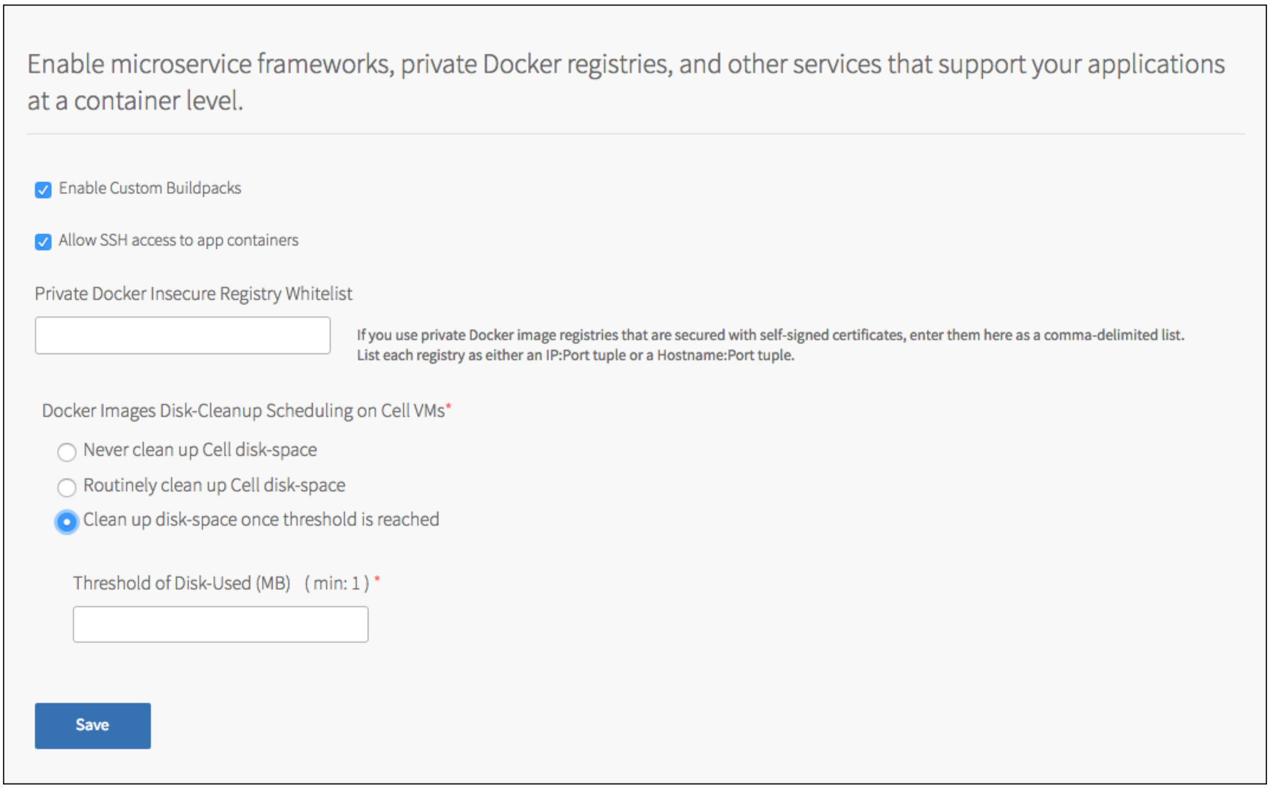
```
$> docker pull openjdk:7
$> docker pull nginx:1.13
$> docker pull php:7-apache


$> docker images
REPOSITORY      TAG        IMAGE ID        CREATED          SIZE
php          7-apache    2720c02fc079    2 days ago        391 MB
openjdk        7        f45207c01009    5 days ago       586 MB
nginx         1.13       3448f27c273f    5 days ago       109 MB
...
```

3. Calculate the threshold as the sum of the frequently used image sizes plus a 15-20% buffer.

   For example, using the output above, the sample threshold calculation is: (391 MB + 586 MB + 109 MB) * 1.2 = 1303.2 MB

Configure Disk Cleanup Scheduling

1. Navigate to the PCF Operations Manager **Installation Dashboard**.
2. Click the Pivotal Application Service (PAS) tile, and navigate to the **Application Containers** tab.



3. Under **Docker Images Disk-Cleanup Scheduling on Cell VMs**, select an option.
4. If you select **Clean up Disk space once threshold is reached**, you must complete the **Threshold of Disk Used (MB)** field. Enter the disk space threshold amount in MB that you calculated for your deployment, as described in Calculating a Threshold.
5. Click **Save**.

Next Steps

If you are configuring PAS for the first time, return to your specific IaaS configuration to continue the installation process.

If you are modifying an existing PAS installation, return to the **PCF Ops Manager Installation Dashboard** and click **Apply Changes**.

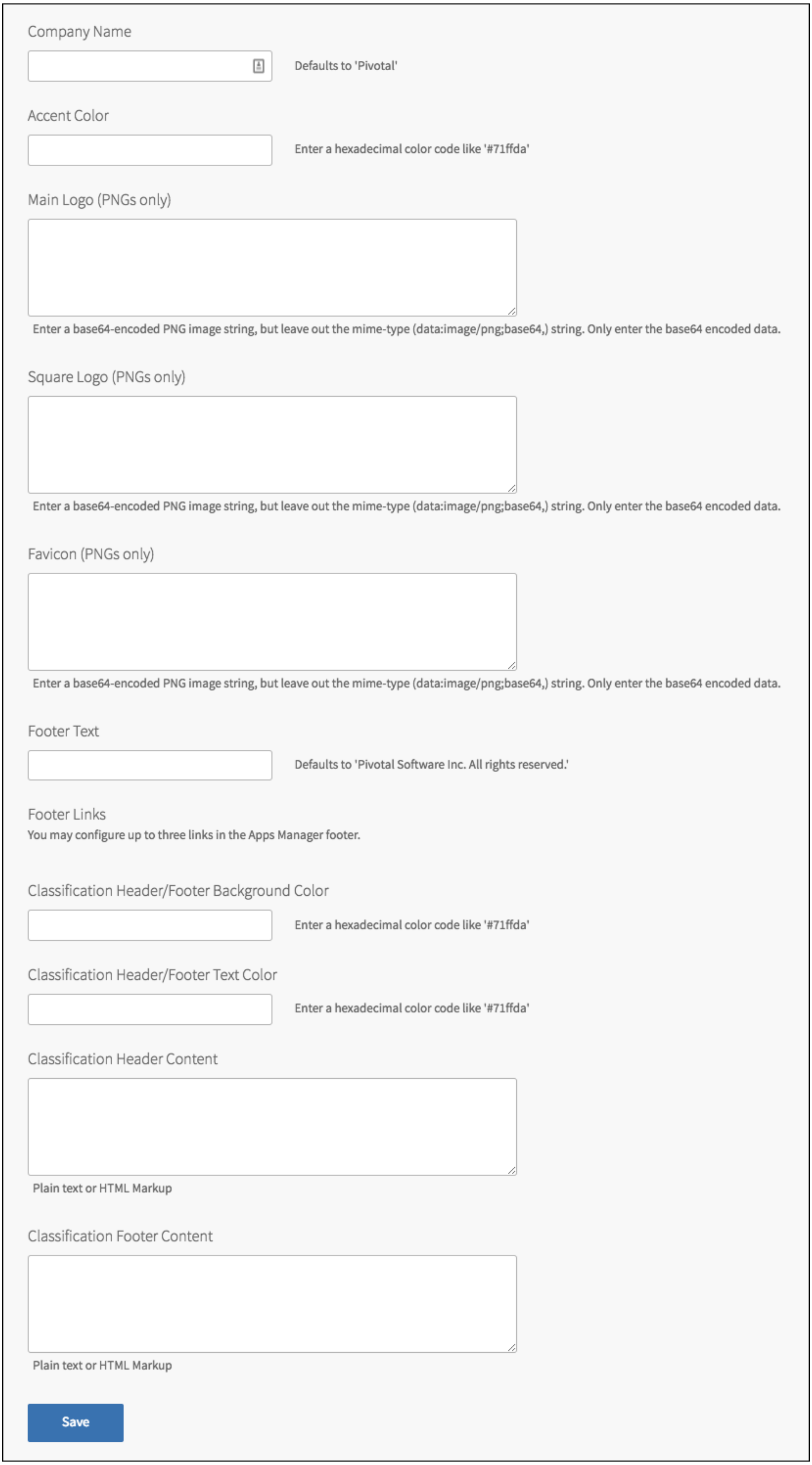
Custom Branding Apps Manager

This topic describes how Pivotal Cloud Foundry operators can visually brand Apps Manager by changing certain text, colors, and images of the interface. Developers view the customized interface when logging in, creating an account, resetting a password, or using Apps Manager.

Operators customize Apps Manager by configuring the **Custom Branding** and **Apps Manager Config** pages of the Pivotal Application Service (PAS) tile.

Custom Branding Page

1. In a browser, navigate to the fully qualified domain name (FQDN) of your Ops Manager and log in.
2. Click **Pivotal Application Service**.
3. Click the **Custom Branding** tab.


Company Name
Defaults to 'Pivotal'
Accent Color
Enter a hexadecimal color code like '#71ffda'
Main Logo (PNGs only)
Enter a base64-encoded PNG image string, but leave out the mime-type (data:image/png;base64,) string. Only enter the base64 encoded data.
Square Logo (PNGs only)
Enter a base64-encoded PNG image string, but leave out the mime-type (data:image/png;base64,) string. Only enter the base64 encoded data.
Favicon (PNGs only)
Enter a base64-encoded PNG image string, but leave out the mime-type (data:image/png;base64,) string. Only enter the base64 encoded data.
Footer Text
Defaults to 'Pivotal Software Inc. All rights reserved.'
Footer Links
You may configure up to three links in the Apps Manager footer.
Classification Header/Footer Background Color
Enter a hexadecimal color code like '#71ffda'
Classification Header/Footer Text Color
Enter a hexadecimal color code like '#71ffda'
Classification Header Content
Plain text or HTML Markup
Classification Footer Content
Plain text or HTML Markup
Save

4. For **Company Name**, enter the name of your organization. If left blank, the name defaults to **Pivotal**.

5. For **Accent Color**, enter the hexadecimal code for the color used to accent various visual elements, such as the currently selected space in the sidebar. For example, `#71ffda`.

6. For **Main Logo**, enter a Base64-encoded URL string for a PNG image to use as your main logo. The image can be square or wide. For example, `data:image/png;base64,iVBORw0...`. If left blank, the image defaults to the Pivotal Logo.

7. For **Square Logo**, enter a Base64-encoded URL string for a PNG image to use in the Apps Manager header and in places that require a smaller logo. For example, `data:image/png;base64,iVBORw0...`. If left blank, the image defaults to the Pivotal Logo.

8. For **Favicon**, enter a Base64-encoded URL string for a PNG image to use as your favicon. For example, `data:image/png;base64,iVBORw0...`. If left blank, the image defaults to the Pivotal Logo.

9. For **Footer Text**, enter a string to be displayed as the footer. If left blank, the footer text defaults to **Pivotal Software Inc. All rights reserved.**.

10. To add up to three footer links that appear to the right of the footer text, complete the following steps:
    - Click **Add**.
    - For **Link text**, enter a label for the link.
    - For **Url**, enter an external or relative URL. For example, `http://docs.pivotal.io` or `/tools.html`.

11. For special notification purposes such as governmental or restricted usage, use the Classification fields to create a special Header and Footer. Enter values in the following fields:
    - For **Classification Header/Footer Background Color**, enter the hexadecimal code for the desired background color of the header and footer.
    - For **Classification Header/Footer Text Color**, enter the hexadecimal code for the desired color of header and footer text.
    - For **Classification Header Content**, enter content for the header in either plain text or HTML. If you enter HTML content, eliminate white spaces and new lines. If you do not provide any content, the custom header will not appear.
    - For **Classification Footer Content**, enter content for the footer in either plain text or HTML. If you enter HTML content, eliminate white spaces and new lines. If you do not provide any content, the custom footer will not appear. The Classification footer appears below the normal footer, which you can customize in the **Footer Text** and **Footer Links** fields.

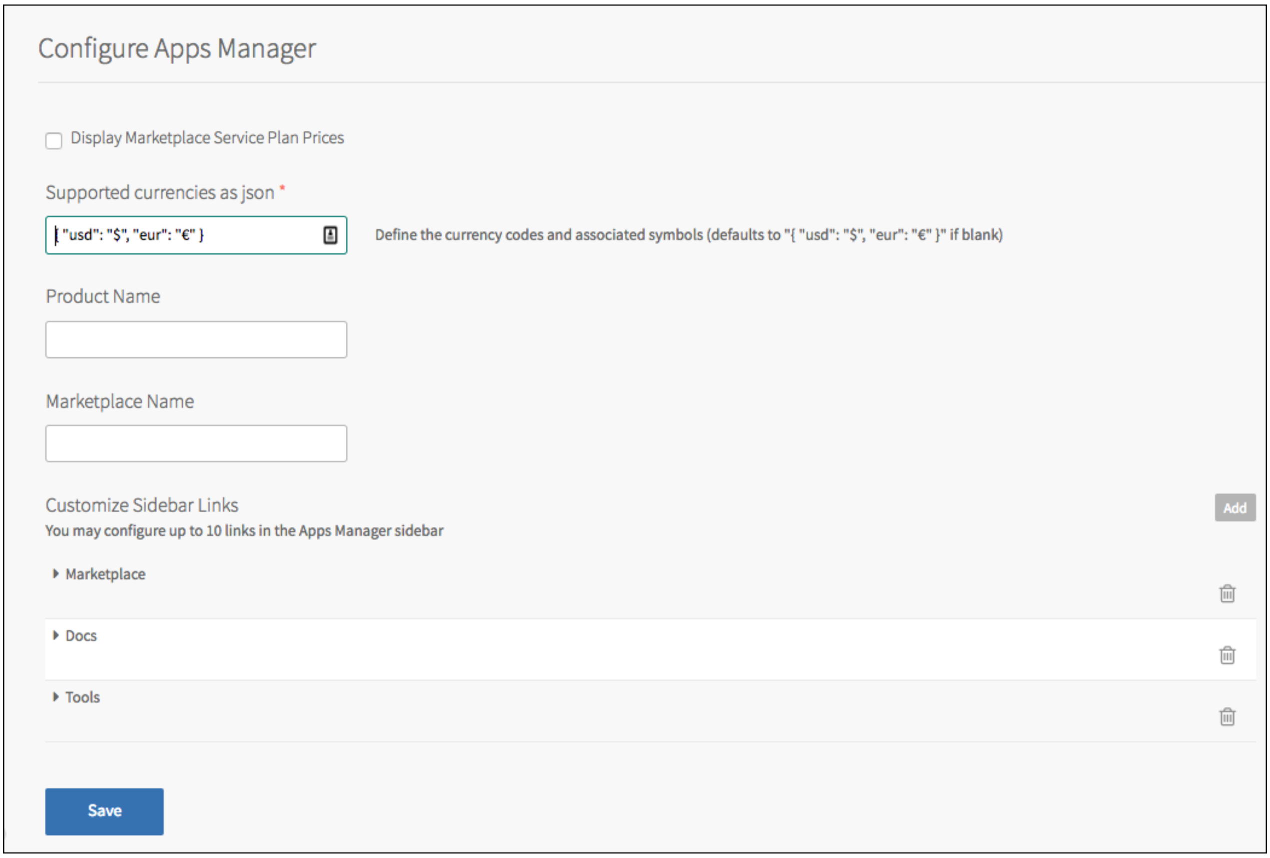
    Note: The Header and Footer do not appear on the User Account and Authentication (UAA) login page.

Apps Manager Config Page

1. In a browser, navigate to the fully qualified domain name (FQDN) of your Ops Manager and log in.

2. Click **Pivotal Application Service**.

3. Click the **Apps Manager Config** tab.



4. For **Product Name**, enter text to replace **Apps Manager** in the header and the title of Apps Manager. This text defaults to **Apps Manager** if left blank.

5. For **Marketplace Name**, enter text to replace the header in the Marketplace pages. This text defaults to **Marketplace** if left blank.

6. By default, Apps Manager includes three sidebar links: **Marketplace**, **Docs**, and **Tools**. You can edit existing sidebar links by clicking the name of the link and editing the **Link text** and **Url** fields. Or, you can remove the link by clicking the trash icon next to its name. If you want to add a new sidebar link, click **Add** and complete the **Link text** and **Url** fields.

> **Note:** Removing any of the default links will remove them from the sidebar for all users.

Reporting App, Task, and Service Instance Usage

Page last updated:

This topic describes how to use the Cloud Foundry Command Line Interface (cf CLI) to retrieve system and org level usage information about your apps, tasks, and service instances through the Cloud Controller and Usage Service APIs.

You can also access usage information by using Apps Manager. For more information, see the Monitoring Instance Usage with Apps Manager topic.

Target and Log In

To perform the procedures in this topic, you must target the endpoint of your Cloud Controller and log in. If you do not have the cf CLI installed, see the Installing the cf CLI topic.

1. Target the endpoint of your Cloud Controller:

```
$ cf api api.YOUR-SYSTEM-DOMAIN
```

2. Log in as admin or as an Org Manager or Org Auditor for the org you want to obtain information about.

```
$ cf login -u admin
API endpoint:  api.YOUR-SYSTEM-DOMAIN
Email: user@example.com
Password:
Authenticating...
OK
...
Targeted org  YOUR-ORG
Targeted space  development
API endpoint:    https://api.YOUR-SYSTEM-DOMAIN (API version:  2.52.0)
User:           user@example.com
Org:            YOUR-ORG
Space:          development
```

Obtain System Usage Information

You can obtain the following system usage information:

- App Usage
- Task Usage
- Service Usage

App Usage

Use `curl` to make a request to `/system_report/app_usages` on the Usage Service endpoint to show system-wide app usage data:

```
$ curl "https://app-usage.YOUR-SYSTEM-DOMAIN/system_report/app_usages" -k -v \
-H "authorization: `cf oauth-token`"
{
"report_time": "2017-04-11 22:28:24 UTC",
"monthly_reports": [
  {
    "month": 4,
    "year": 2017,
    "average_app_instances": 17855.256153713308,
    "maximum_app_instances": 18145,
    "app_instance_hours": 4686533.080277303
  }
 ]
"yearly_reports": [
  {
    "year": 2017,
    "average_app_instances": 16184.9,
    "maximum_app_instances": 18145,
    "app_instance_hours": 39207433.0802773
  }
 ]
}
```

Task Usage

Use `curl` to make a request to `/system_report/task_usages` on the Usage Service endpoint to show system-wide task usage data:

```
$ curl "https://app-usage.YOUR-SYSTEM-DOMAIN/system_report/task_usages" -k -v \
-H "authorization: `cf oauth-token`"
{
"report_time": "2017-04-11T22:33:48.971Z",
"monthly_reports": [
  {
    "month": 4,
    "year": 2017,
    "total_task_runs": 235,
    "maximum_concurrent_tasks": 7,
    "task_hours": 43045.201944444445
  }
 ]
"yearly_reports": [
  {
    "year": 2017,
    "total_task_runs": 2894,
    "maximum_concurrent_tasks": 184,
    "task_hours": 45361.26694444445
  }
 ]
}
```

Service Usage

Use `curl` to make a request to `/system_report/service_usages` on the Usage Service endpoint to show system-wide service usage data:

```
$ curl "https://app-usage.YOUR-SYSTEM-DOMAIN/system_report/service_usages" -k -v \
-H "authorization: `cf oauth-token`"
{
 "report_time": "2017-05-11 18:29:14 UTC",
 "monthly_service_reports": [
  {
    "service_name": "fake-service-0507f1fd-2340-49a6-9d43-a347a5f5f6be",
    "service_guid": "177dcfde-cd51-4058-bd86-b98015c295f5",
    "usages": [
      {
       "month": 1,
       "year": 2017,
       "duration_in_hours": 0,
       "average_instances": 0,
       "maximum_instances": 0
      },
      {
       "month": 2,
       "year": 2017,
       "duration_in_hours": 0,
```

```
      "average_instances": 0,
      "maximum_instances": 0
    },
    {
      "month": 3,
      "year": 2017,
      "duration_in_hours": 4.182222222222227,
      "average_instances": 0,
      "maximum_instances": 2
    },
    {
      "month": 4,
      "year": 2017,
      "duration_in_hours": 2176.962222222186,
      "average_instances": 3,
      "maximum_instances": 7
    },
    {
      "month": 5,
      "year": 2017,
      "duration_in_hours": 385.61388888888854,
      "average_instances": 1.5,
      "maximum_instances": 3
    }
  ],
  "plans": [
    {
      "usages": [
        {
          "month": 1,
          "year": 2017,
          "duration_in_hours": 0,
          "average_instances": 0,
          "maximum_instances": 0
        },
        {
          "month": 2,
          "year": 2017,
          "duration_in_hours": 0,
          "average_instances": 0,
          "maximum_instances": 0
        },
        {
          "month": 3,
          "year": 2017,
          "duration_in_hours": 4.182222222222227,
          "average_instances": 0,
          "maximum_instances": 2
        },
        {
          "month": 4,
          "year": 2017,
          "duration_in_hours": 1465.6388888888941,
          "average_instances": 2,
          "maximum_instances": 5
        },
        {
          "month": 5,
          "year": 2017,
          "duration_in_hours": 385.61388888888854,
          "average_instances": 1.5,
          "maximum_instances": 3
        }
      ],
      "service_plan_name": "fake-plan",
      "service_plan_guid": "ac09f607-f4e5-4807-af16-e95856061bd7"
    }
```


Obtain Org Usage Information

You can obtain the following org usage information:

- App Usage
- Task Usage
- Service Usage

You must have the GUID of the org you want to obtain information about in order to perform the procedures in this section. To retrieve your org GUID, run the `cf org` command:

```
$ cf org YOUR-ORG --guid
```

App Usage

Use `curl` to make a request to `/app_usages` on the Usage Service endpoint to show app usage in an org. You must complete the placeholders in `start=YYYY-MM-DD&end=YYYY-MM-DD` to define a date range.

```
$ curl "https://app-usage.YOUR-SYSTEM-DOMAIN/organizations/YOUR-ORG-GUID/app_usages?start=YYYY-MM-DD&end=YYYY-MM-DD" \
-k -v \
-H "authorization: `cf oauth-token`"
{
    "organization_guid": "8d83362f-587a-4148-806b-4407428887b5",
   "period_start": "2016-06-01T00:00:00Z",
   "period_end": "2016-06-13T23:59:59Z",
   "app_usages": [
     {
      space_guid: "44435fd6-fbac-5049-bbfc-92d1603a5e98"
      space_name: "outer-space"
      app_guid: "00ecd7ce-1dd0-4b3f-63b9-744c9de42afc"
      app_name: "your-app"
      instance_count: 6
      memory_in_mb_per_instance: 64
      duration_in_seconds: 76730
     }
   ]
  }
```

Task Usage

Use `curl` to make a request to `/task_usages` on the Usage Service endpoint to show task usage in an org. You must complete the placeholders in `start=YYYY-MM-DD&end=YYYY-MM-DD` to define a date range.

```
$ curl "https://app-usage.YOUR-SYSTEM-DOMAIN/organizations/YOUR-ORG-GUID/task_usages?start=YYYY-MM-DD&end=YYYY-MM-DD" \
-k -v \
-H "authorization: `cf oauth-token`"
{
   "organization_guid": "8f88362f-547c-4158-808b-4605468387d5",
   "period_start": "2014-01-01",
   "period_end": "2017-04-04",
   "spaces": {
    "e6445eb3-fdac-4049-bafc-94d1703d5e78": {
     "space_name": "smoketest",
     "task_summaries": [
      {
       "parent_application_guid": "04cd29d5-1f9e-4900-ac13-2e903f6582a9",
       "parent_application_name": "task-dummy-app",
       "memory_in_mb_per_instance": 256,
       "task_count_for_range": 54084,
       "concurrent_task_count_for_range": 5
       "total_duration_in_seconds_for_range": 37651415
      },
     ]
    },
    "b66665e4-873f-4482-acf1-89307ba9c6e4": {
     "space_name": "smoketest-experimental",
     "task_summaries": [
     {
       "parent_application_guid": "d941b689-4a27-44ec-91d3-1f97434dbed9",
       "parent_application_name": "console-blue",
       "memory_in_mb_per_instance": 256,
       "task_count_for_range": 14,
       "concurrent_task_count_for_range": 2
       "total_duration_in_seconds_for_range": 20583
      }
     ]
    }
   }
}
```

Note: In the `/task_usages` endpoint, `memory_in_mb_per_instance` is the memory of the task.

Service Usage

Use `curl` to make a request to `/service_usages` on the Usage Service endpoint to show service usage in an org. You must complete the placeholders in `start=YYYY-MM-DD&end=YYYY-MM-DD` to define a date range.

```
$ curl "https://app-usage.YOUR-SYSTEM-DOMAIN/organizations/`cf org YOUR-ORG --guid`/service_usages?start=YYYY-MM-DD&end=YYYY-MM-DD" -k -v -H "authorization: `cf oauth
 {
   "organization_guid": "8d83362f-587a-4148-806b-4407428887b5",
   "period_start": "2016-06-01T00:00:00Z",
   "period_end": "2016-06-13T23:59:59Z",
   "service_usages": [
     {
       deleted: false
       duration_in_seconds: 1377982.52
       service_guid: "02802293-b769-44cc-807f-eee331ba9b2b"
       service_instance_creation: "2016-01-20T18:48:16.000Z"
       service_instance_deletion: null
       service_instance_guid: "b25b4481-19aa-4504-82c9-f303e7e9ed6e"
       service_instance_name: "something-usage-db"
       service_instance_type: "managed_service_instance"
       service_name: "my-mysql-service"
       service_plan_guid: "70915a70-7311-4f3e-ab0d-4a7dfd3ef2d9"
       service_plan_name: "20gb"
       space_guid: "e6445eb3-fdac-4049-bafc-94d1703d5e78"
       space_name: "outer-space"
     }
   ]
 }
```

Obtain App Information

You can obtain the following app information:

- All Apps
- App Summary

All Apps

Use the `apps` endpoint to retrieve information about all of your apps:

```
$ cf curl /v2/apps
{
 "total_results": 2,
 "total_pages": 1,
 "prev_url": null,
 "next_url": null,
 "resources": [
   {
      "metadata": {
        "guid": "acf2ce33-ee92-54TY-9adb-55a596a8dcba",
        "url": "/v2/apps/acf2ce33-ee92-54TY-9adb-55a596a8dcba",
        "created_at": "2016-02-06T17:40:31Z",
        "updated_at": "2016-02-06T18:09:17Z"
      },
      "entity": {
        "name": "YOUR-APP",
 [...]
        "space_url": "/v2/spaces/a0205ae0-a691-4667-92bc-0d0dd712b6d3",
        "stack_url": "/v2/stacks/86205f38-84fc-4bc2-b2b8-af7f55669f04",
        "routes_url": "/v2/apps/acf2ce33-ee92-54TY-9adb-55a596a8dcba/routes",
        "events_url": "/v2/apps/acf2ce33-ee92-54TY-9adb-55a596a8dcba/events",
        "service_bindings_url": "/v2/apps/acf2ce33-ee92-54TY-9adb-55a596a8dcba/service_bindings",
       "route_mappings_url": "/v2/apps/acf2ce33-ee92-54TY-9adb-55a596a8dcba/route_mappings"
        }
      },
   {
      "metadata": {
        "guid": "79bb58cc-3737-43be-ac70-39a2843b5178",
        "url": "/v2/apps/79bb58cc-3737-4540-ac70-39a2843b5178",
        "created_at": "2016-02-15T23:25:47Z",
        "updated_at": "2016-03-12T21:54:59Z"
      },
      "entity": {
        "name": "ANOTHER-APP",
[...]
        "space_url": "/v2/spaces/a0205ae0-a691-4667-92bc-0d0dd712b6d3",
        "stack_url": "/v2/stacks/86205f38-84fc-4bc2-b2b8-af7f55669f04",
        "routes_url": "/v2/apps/79bb58cc-3737-4540-ac70-39a2843b5178/routes",
        "events_url": "/v2/apps/79bb58cc-3737-4540-ac70-39a2843b5178/events",
        "service_bindings_url": "/v2/apps/79bb58cc-3737-4540-ac70-39a2843b5178/service_bindings",
        "route_mappings_url": "/v2/apps/79bb58cc-3737-4540-ac70-39a2843b5178/route_mappings"
       }
     }
   ]
}
```

The output of this command provides the URL endpoints for each app, within the `metadata: url` section. You can use these app-specific endpoints to retrieve more information about that app. In the example above, the endpoints for the two apps are `/v2/apps/acf2ce33-ee92-54TY-9adb-55a596a8dcba` and `/v2/apps/79bb58cc-3737-4540-ac70-39a2843b5178`.

App Summary

Use the `summary` endpoint under each app-specific URL to retrieve the instances and any bound services for that app:

```
$ cf curl /v2/apps/acf2ce75-ee92-4bb6-9adb-55a596a8dcba/summary
{
    "guid": "acf2ce75-ee92-4bb6-9adb-55a596a8dcba",
  "name": "YOUR-APP",
  "routes": [
  {
    "guid": "7421b6af-75cb-4334-a862-bc5e1ababfe6",
    "host": "YOUR-APP",
    "path": "",
    "domain": {
      "guid": "fb6bd89f-2ed9-49d4-9ad1-97951a573135",
      "name": "YOUR-SYSTEM-DOMAIN.io"
    }
  }
],
 "running_instances": 5,
 "services": [
   {
     "guid": "b9cdr456-3c61-4f8a-a307-9b4ty836fb2e",
     "name": "YOUR-APP-db",
     "bound_app_count": 1,
     "last_operation": {
       "type": "create",
       "state": "succeeded",
       "description": "",
       "updated_at": null,
       "created_at": "2016-02-05T04:58:46Z"
   },
     "dashboard_url": "https://cloudfoundry.appdirect.com/api/custom/cloudfoundry/v2/sso/start?serviceUuid=b5cASDF-3c61-4f8a-a307-9bf85j45fb2e",
     "service_plan": {
       "guid": "fbcec3af-3e8d-4ee7-adfe-3f12a137ed66",
       "name": "turtle",
       "service": {
         "guid": "34dbc753-34ed-4cf1-9a87-a255dfca5339b",
         "label": "elephantsql",
         "provider": null,
         "version": null
[...]
```

To view the `app_usages` report that covers app usage within an org during a period of time, see Obtain Usage Information About an Org.

Obtain Services Information

Use the `service_instances?` endpoint to retrieve information about both bound and unbound service instances:

```
$ cf curl /v2/service_instances?
{
  "total_results": 4,
  "total_pages": 1,
  "prev_url": null,
  "next_url": null,
  "resources": [
    {
    "metadata": {
       "guid": "b9cdr456-3c61-4f8a-a307-9b4ty836fb2e",
       "url": "/v2/service_instances/b9cdr456-3c61-4f8a-a307-9b4ty836fb2e",
       "created_at": "2016-02-05T04:58:46Z",
       "updated_at": null
    },
    "entity": {
       "name": "YOUR-BOUND-DB-INSTANCE",
       "credentials": {},
       "service_plan_guid": "fbcec3af-3e8d-4ee7-adfe-3f12a137ed66",
       "space_guid": "a0205ae0-a691-4667-92bc-0d0dd712b6d3",
       "gateway_data": null,
       "dashboard_url": "https://cloudfoundry.appdirect.com/api/custom/cloudfoundry/v2/sso/start?serviceUuid=b9cdr456-3c61-4f8a-a307-9b4ty836fb2e",
       "type": "managed_service_instance",
       "last_operation": {
          "type": "create",
          "state": "succeeded",
          "description": "",
          "updated_at": null,
          "created_at": "2016-02-05T04:58:46Z"
        },
        "tags": [],
        "space_url": "/v2/spaces/a0205ae0-a691-4667-92bc-0d0dd712b6d3",
        "service_plan_url": "/v2/service_plans/fbcec3af-3e8d-4ee7-adfe-3f12a137ed66",
        "service_bindings_url": "/v2/service_instances/b9cdr456-3c61-4f8a-a307-9b4ty836fb2e/service_bindings",
        "service_keys_url": "/v2/service_instances/b9cdr456-3c61-4f8a-a307-9b4ty836fb2e/service_keys",
        "routes_url": "/v2/service_instances/b9cdr456-3c61-4f8a-a307-9b4ty836fb2e/routes"
      }
    },
    {
    "metadata": {
       "guid": "78be3399-bdc7-4fbf-a1a4-6858a50d0ff3",
       "url": "/v2/service_instances/78be3399-bdc7-4fbf-a1a4-6858a50d0ff3",
       "created_at": "2016-02-15T23:45:30Z",
       "updated_at": null
    },
    "entity": {
       "name": "YOUR-UNBOUND-DB-INSTANCE",
       "credentials": {},
       "service_plan_guid": "fbcec3af-3e8d-4ee7-adfe-3f12a137ed66",
       "space_guid": "a0205ae0-a691-4667-92bc-0d0dd712b6d3",
       "gateway_data": null,
       "dashboard_url": "https://cloudfoundry.appdirect.com/api/custom/cloudfoundry/v2/sso/start?serviceUuid=78be3399-bdc7-4fbf-a1a4-6858a50d0ff3",
       "type": "managed_service_instance",
       "last_operation": {
         "type": "create",
         "state": "succeeded",
         "description": "",
         "updated_at": null,
         "created_at": "2016-02-15T23:45:30Z"
         },
       "tags": [],
       "space_url": "/v2/spaces/a0205ae0-a691-4667-92bc-0d0dd712b6d3",
       "service_plan_url": "/v2/service_plans/fbcec3af-3e8d-4ee7-adfe-3f12a137ed66",
       "service_bindings_url": "/v2/service_instances/78be3399-bdc7-4fbf-a1a4-6858a50d0ff3/service_bindings",
       "service_keys_url": "/v2/service_instances/78be3399-bdc7-4fbf-a1a4-6858a50d0ff3/service_keys",
       "routes_url": "/v2/service_instances/78be3399-bdc7-4fbf-a1a4-6858a50d0ff3/routes"
       }
     },
   ]
 }
```

Reporting Instance Usage with Apps Manager

Page last updated:

This topic describes how to retrieve app, task, and service instance usage information using Apps Manager.

You can also retrieve app, task, and service instance usage information using the Usage service API, or the Cloud Foundry API from the Cloud Foundry Command Line Interface (cf CLI). For more information, see the Monitoring App, Task, and Service Instance Usage topic.

There are two ways to monitor app, task, and service instance usage from Apps Manager. The Accounting Report provides a summarized report, and the Usage Report provides a more detailed view of the data.

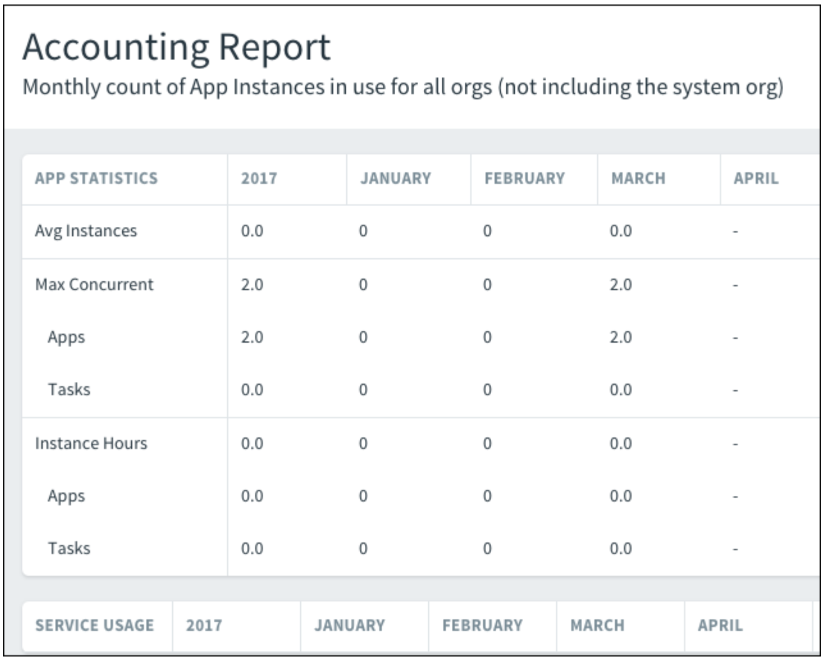
View the Accounting Report

The Accounting Report displays instance usage information for all orgs in your Pivotal Cloud Foundry (PCF) deployment except the **system** org. The Accounting Report provides a high-level overview of your usage by displaying your monthly count of app, task, and service instances.

Follow the steps below to access the Accounting Report.

1. Log into the Apps Manager as an admin.
2. From the left navigation of Apps Manager, select **Accounting Report**.
3. Under **App Statistics** and **Service Usage**, view the average and maximum instances in use per month.

   Max Concurrent displays the largest number of instances in use at a time over the indicated time period. The Accounting Report calculates these values from the `start`, `stop`, and `scale` app usage events in Cloud Controller.



APP STATISTICS	2017	JANUARY	FEBRUARY	MARCH	APRIL
Avg Instances	0.0	0	0	0.0	-
Max Concurrent	2.0	0	0	2.0	-
Apps	2.0	0	0	2.0	-
Tasks	0.0	0	0	0.0	-
Instance Hours	0.0	0	0	0.0	-
Apps	0.0	0	0	0.0	-
Tasks	0.0	0	0	0.0	-

SERVICE USAGE	2017	JANUARY	FEBRUARY	MARCH	APRIL

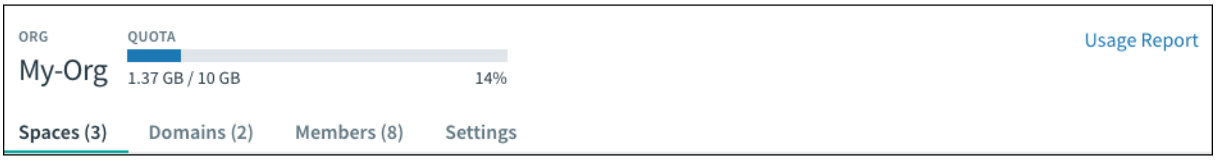
View the Usage Report

The Usage Report provides a more granular view of your usage by detailing app, task, and service instance usage information for all spaces in a particular org, excluding the **system** org.

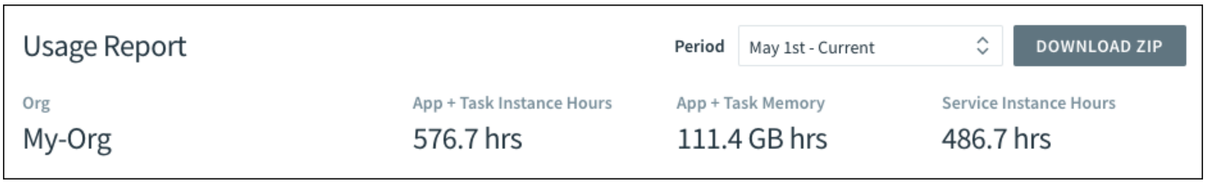
Follow the steps below to access the Usage Report.

1. Log into the Apps Manager as an admin, or as an account with the **Org Manager** or **Org Auditor** role. For more information about managing roles, see the Managing User Accounts and Permissions Using the Apps Manager topic.
2. From the dropdown on the left, select the org for which you want to view a usage report.

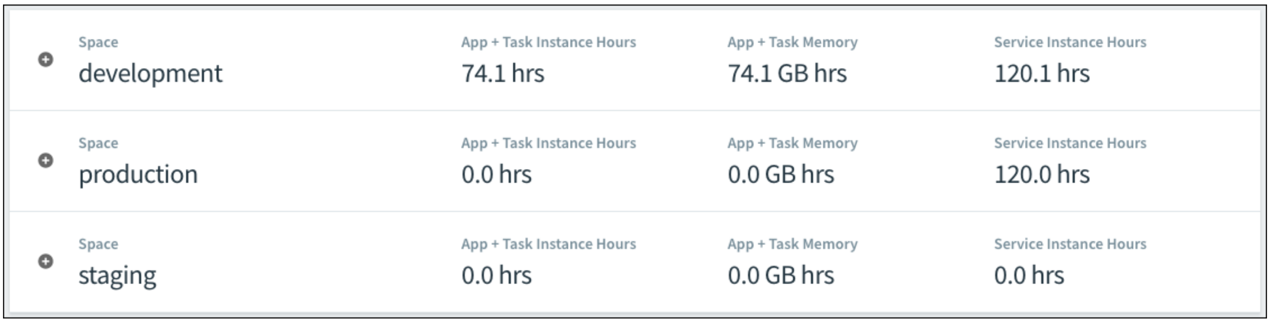
3. Click **Usage Report** in the upper right.



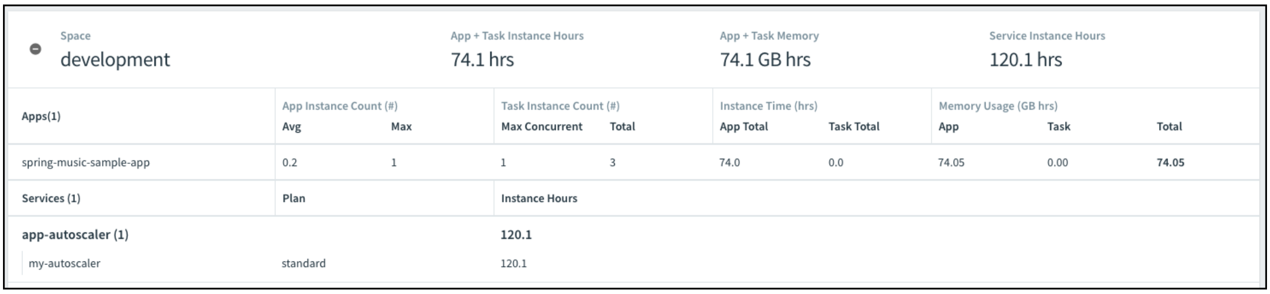
The top of the Usage Report displays total **App + Task Instance Hours**, **App + Task Memory**, and **Service Instance Hours** by all spaces in the org.



The report provides total usage information for each of your spaces.



To display more detailed information about app, task, and service instance usage in a space, click the name of the space for an expanded view.



Providing a Certificate for Your TLS Termination Point

Page last updated:

This topic describes how to configure Transport Layer Security (TLS) termination for HTTP traffic in Pivotal Application Service (PAS) with a TLS certificate, as part of the process of configuring PAS for deployment.

Configure TLS Termination

When you deploy PCF, you must configure the TLS termination for HTTP traffic in your PAS configuration. You can terminate TLS at all of the following points:

- Load Balancer
- Load Balancer and Gorouter
- Gorouter

Follow the guidance in Securing Traffic into Cloud Foundry to choose and configure the TLS termination option for your deployment.

> **Note**: If you are using HAProxy in a PCF deployment, you can choose to terminate SSL/TLS at HAProxy in addition to any of the SSL/TLS termination options above. For more information, see Configuring SSL/TLS Termination at HAProxy.

Obtain TLS Certificates

To secure traffic into PCF, you must obtain at least one TLS certificate. See Certificate Requirements for general certificate requirements for deploying PCF.

See the following sections for additional IaaS-specific certificate requirements:

- **AWS**: Certificate Requirements on AWS
- **GCP**: Certificate Requirements on GCP

Creating a Wildcard Certificate for PCF Deployments

This section describes how to create or generate a certificate for your PAS environment. If you are deploying to a production environment, you should obtain a certificate from a trusted authority (CA).

For internal development or testing environments, you have two options for creating a required TLS certificates.

- You can create a self-signed certificate, or
- You can have PAS generate the certificate for you.

To create a certificate, you can use a wide variety of tools including OpenSSL, Java’s keytool, Adobe Reader, and Apple’s Keychain to generate a Certificate Signing Request (CSR).

In either case for either self-signed or trusted single certificates, apply the following rules when creating the CSR:

- Specify your registered wildcard domain as the `Common Name`. For example, `*.yourdomain.com`.
- If you are using a split domain setup that separates the domains for `apps` and `system` components (recommended), then enter the following values in the `Subject Alternative Name` of the certificate:
  - `*.apps.yourdomain.com`
  - `*.system.yourdomain.com`
  - `*.login.system.yourdomain.com`
  - `*.uaa.system.yourdomain.com`
- If you are using a single domain setup, then use the following values as the `Subject Alternative Name` of the certificate:
  - `*.login.system.yourdomain.com`

- `*.uaa.system.yourdomain.com`

> **Note**: TLS certificates generated for wildcard DNS records only work for a single domain name component or component fragment. For example, a certificate generated for `*.EXAMPLE.com` does not work for `*.apps.EXAMPLE.com` and `*.system.EXAMPLE.com`. The certificate must have both `*.apps.EXAMPLE.com` and `*.system.EXAMPLE.com` attributed to it.

Generating a RSA Certificate in PAS

1. Navigate to the Ops Manager Installation Dashboard.

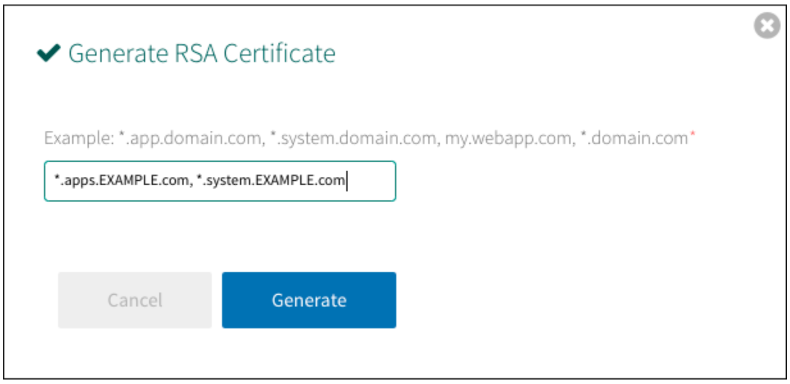
2. Click the **Pivotal Application Service** tile in the Installation Dashboard.

3. Click **Networking**.

4. Click **Generate RSA Certificate** to populate the **Certificate and Private Key for HAProxy and Router** fields with RSA certificate and private key information.

5. If you are using a split domain setup that separates the domains for `apps` and `system` components (recommended), then enter the following domains for the certificate:
   - `*.yourdomain.com`
   - `*.apps.yourdomain.com`
   - `*.system.yourdomain.com`
   - `*.login.system.yourdomain.com`
   - `*.uaa.system.yourdomain.com`

   For example, `*.example.com`, `*.apps.example.com`, `*.system.example.com`, `*.login.system.example.com`, `*.uaa.system.example.com`



Enabling NFS Volume Services

This topic describes how Pivotal Cloud Foundry (PCF) operators can deploy NFS volume services.

Overview

A volume service gives apps access to a persistent filesystem, such as NFS. By performing the procedures in this topic, operators can add a volume service to the Marketplace that provides an NFSv3 filesystem.

Developers can then use the Cloud Foundry Command Line Interface (cf CLI) to create service instances of the volume service and bind them to their apps. For more information, see the Using an External File System (Volume Services) topic.

> **Note**: You must have an NFS server running to enable NFS volume services. If you want to use NFS and do not currently have a server available, you can deploy the test NFS server bundled with the NFS Volume release.

Enable Volume Services

To enable NFS volume services in PCF, perform the following steps:

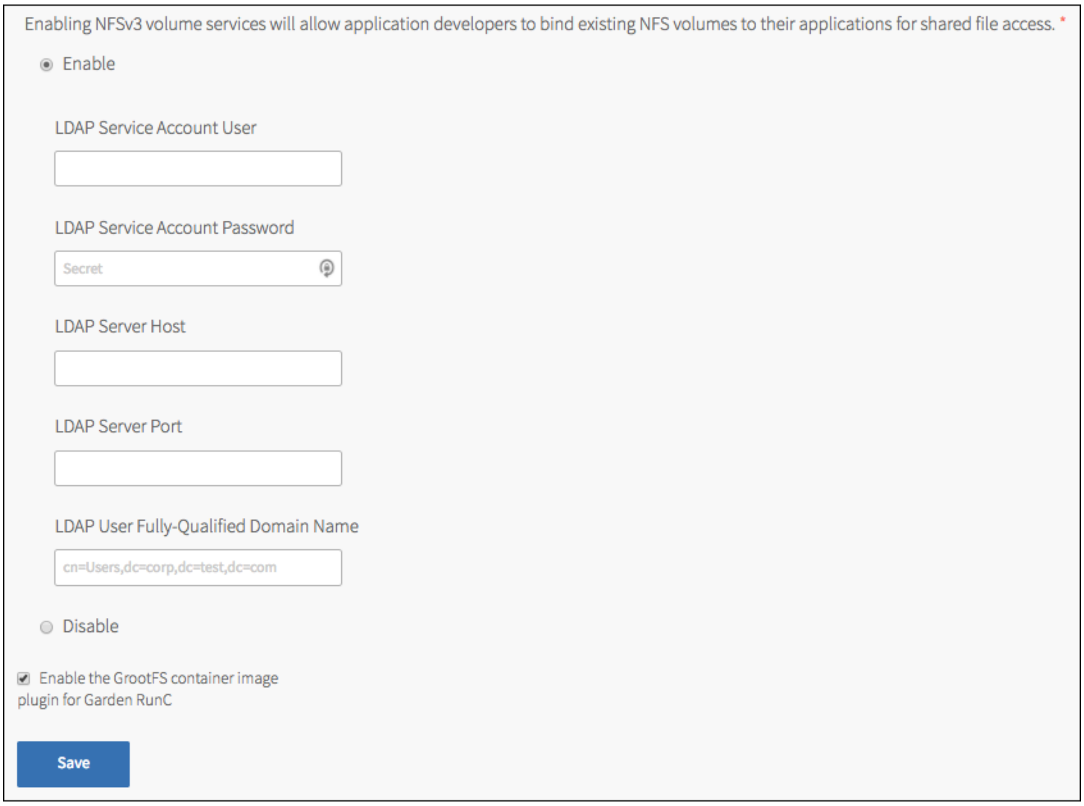
1. Navigate to the Ops Manager Installation Dashboard.

2. Click the **Pivotal Application Service** tile.

3. Click **Application Containers**.

4. Under **Enabling NFSv3 volume services**, select **Enable**.

    > **Note**: In a clean install, NFSv3 volume services are enabled by default. In an upgrade, NFSv3 volume services match the setting of the previous deployment.

5. (Optional) To configure LDAP for NFSv3 volume services, perform the following steps:

- For **LDAP Service Account User**, enter the username of the service account in LDAP that will manage volume services.
- For **LDAP Service Account Password**, enter the password for the service account.
- For **LDAP Server Host**, enter the hostname or IP address of the LDAP server.
- For **LDAP Server Port**, enter the LDAP server port number. If you do not specify a port number, Ops Manager uses 389.
- For **LDAP Server Protocol**, enter the server protocol. If you do not specify a protocol, Ops Manager uses TCP.
- For **LDAP User Fully-Qualified Domain Name**, enter the fully qualified path to the LDAP service account. For example, if you have a service account called `volume-services` that belongs to organizational units (OU) called `service-accounts` and `my-company`, and your domain is called `domain`, the fully qualified path looks like the following:

  ```
  CN=volume-services,OU=service-accounts,OU=my-company,DC=domain,DC=com
  ```

6. Click **Save**.

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

8. Using the cf CLI, enable access to the service:

   ```
   $ cf enable-service-access nfs
   ```

   To limit access to a specific org, use the `-o` flag, followed by the name of the org where you want to enable access. For more information, see the Access Control topic.

After completing these steps, developers can use the cf CLI to create service instances of the `nfs` service and bind them to their apps.

Rotating PAS CredHub Encryption Keys

Page last updated:

This topic discusses how to rotate Pivotal Application Service (PAS) CredHub encryption keys. Encryption keys are values that CredHub uses to obscure stored secrets. When an operator marks an additional key as primary, CredHub can rotate in that additional key as the encryption key.

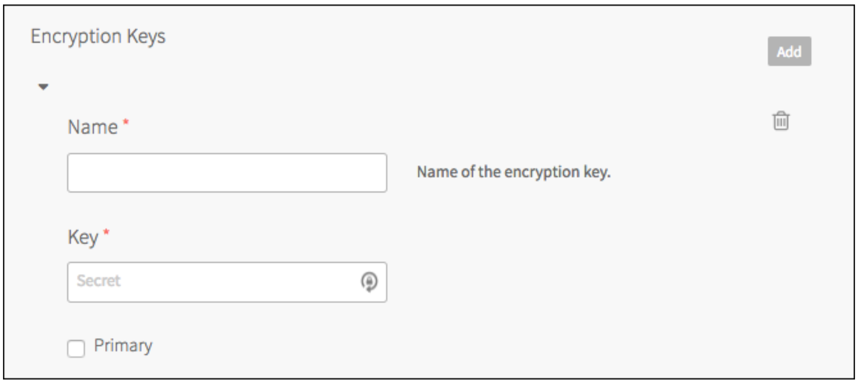
During this credential rotation process, the initial encryption key is used to access the hidden value, That value is then stored again by the additional encryption key.

WARNING: If removes an encryption key and click **Apply Changes** before the rotation completes, the deployment enters a broken state. In this state, you can no longer access data stored with the deleted key.

Rotate PAS encryption keys

Follow the steps below to rotate PAS encryption keys.

1. Navigate to the **Ops Manager Installation Dashboard**.
2. Click the **Pivotal Application Service** tile .
3. Select the **CredHub** tab.



4. In the **Encryption Keys** section, click **Add**.
5. For **Name**, enter the name of your new encryption key.
6. For **Key**, enter your new encryption key.
7. Select the **Primary** check box.
8. Click **Save**.
9. Navigate to the **Ops Manager Installation Dashboard**.
10. Click **Apply Changes**.

Verify PAS encryption key rotation

Follow the steps below to verify that the rotation completes.

1. Click the **Pivotal Application Service** tile.
2. Select the **Status** tab.
3. Within the **CredHub** job, locate **Index 0**.

JOB	INDEX	IPS	AZ	CID	LOAD AVG15	CPU	MEMORY	SWAP	SYSTEM DISK	EPHEM. DISK	PERS. DISK	LOGS
Consul	0	10.0.4.5	us-central1-a	vm-dca0a175-3a56-4cb0-6b64-2d3f0a39ddc7	0.00	1.2%	14%	0%	40%	3%	3%	
NATS	0	10.0.4.6	us-central1-a	vm-a5b7a6e3-d80d-44d2-54cc-a160d3f66916	0.00	1.1%	14%	0%	40%	3%	N/A	
File Storage	0	10.0.4.7	us-central1-a	vm-c4aa5642-54f9-436e-5c31-03b2607de077	0.00	1.0%	4%	0%	40%	7%	3%	
MySQL Proxy	0	10.0.4.8	us-central1-a	vm-c6df54cc-128c-440b-4835-e2826b9502be	0.00	0.3%	15%	0%	40%	4%	N/A	
MySQL Server	0	10.0.4.9	us-central1-a	vm-5ec3aa62-f066-42e0-60ec-ce4464cfd628	0.08	1.1%	24%	0%	40%	3%	5%	
Backup Prepare Node	0	10.0.4.10	us-central1-a	vm-55e93a88-ce00-4519-4cbb-39b12bd82495	0.00	0.3%	19%	0%	40%	20%	0%	
Diego BBS	0	10.0.4.11	us-central1-a	vm-c95a4858-b9fb-460e-4693-1dfd1f3be88d	0.03	1.1%	20%	0%	40%	5%	N/A	
UAA	0	10.0.4.12	us-central1-a	vm-9f5c1fdc-704e-484c-7e89-58c95a19a319	0.02	0.6%	20%	0%	40%	3%	N/A	
Cloud Controller	0	10.0.4.13	us-central1-a	vm-6ec5f788-a390-45ed-4990-9cbd3f6aeee6	0.07	1.0%	27%	0%	40%	16%	N/A	
HAProxy	0	10.0.4.14	us-central1-a	vm-9466dc03-d2bc-48f0-401b-d9082e0f8778	0.00	0.5%	14%	0%	40%	3%	40%	
Router	0	10.0.4.15	us-central1-a	vm-32ba827f-564b-4462-7b20-46a658a8e8df	0.00	1.2%	16%	0%	40%	4%	N/A	
MySQL Monitor	0	10.0.4.16	us-central1-a	vm-98c9c06c-e381-455f-517c-62e9ad9d70d1	0.00	0.2%	11%	0%	40%	2%	N/A	
Clock Global	0	10.0.4.17	us-central1-a	vm-65a93076-941e-4b4c-50a2-e067bfe2d0ed	0.00	0.3%	41%	1%	40%	86%	N/A	
Cloud Controller Worker	0	10.0.4.18	us-central1-a	vm-de45f832-e290-4e38-7692-236fa5b417d5	0.00	1.1%	48%	1%	40%	9%	N/A	
Diego Brain	0	10.0.4.19	us-central1-a	vm-247bcbf3-9241-49df-57fd-7c34d8bb9fef	0.00	0.4%	9%	0%	41%	6%	0%	
Diego Cell	0	10.0.4.20	us-central1-a	vm-99761423-b711-4871-7aa9-848537f5096e	0.07	0.6%	12%	0%	40%	8%	N/A	
Loggregator Trafficcontroller	0	10.0.4.21	us-central1-a	vm-37c47f42-f226-4728-6acc-d4351a7d06cc	0.00	1.4%	18%	0%	40%	4%	N/A	
Syslog Adapter	0	10.0.4.22	us-central1-a	vm-9c577460-1a29-4741-7a0b-e70d6d93b286	0.00	1.1%	14%	0%	40%	3%	N/A	
Syslog Scheduler	0	10.0.4.23	us-central1-a	vm-43a14a07-aaae-466d-6839-23767c3ce141	0.00	0.4%	14%	0%	40%	3%	N/A	
Doppler Server	0	10.0.4.24	us-central1-a	vm-66359c07-0ef4-4a42-542c-2e7cee9e4be5	0.05	1.5%	19%	0%	40%	4%	N/A	
CredHub	0	10.0.4.25	us-central1-a	vm-84b31970-374a-4e0c-411c-3977b6a20ff1	0.00	0.2%	12%	0%	40%	7%	N/A	
	1	10.0.4.26	us-central1-b	vm-b67b89d6-0415-4021-7950-48c6c3aa2b14	0.00	0.2%	12%	0%	40%	7%	N/A	

4. Within the **Logs** column, click the correlating download icon.

5. Select the **Logs** tab.

6. Click the corresponding link to the retrieve the downloaded log file.

7. Unzip the log file.

8. Unzip the larger of the two nested directories.

9. Ops Manager generates a compressed file for each CredHub VM that exists on your deployment. Unzip each of these compressed files.

10. Open the `credhub` directory.

11. Open the `credhub.log` file. If the PAS credential rotation completed successfully, the CredHub log contains the following string: `Successfully rotated NUMBER-OF-CREDENTIALS items`

12. Remove the old encryption key.

13. Click the trashcan icon that corresponds to the old encryption key.

14. Click **Save**.

15. Navigate to the **Ops Manager Installation Dashboard**.

16. Click **Apply Changes**.

Administering and Operating Cloud Foundry

For Administrators of a Running Cloud Foundry Deployment

- Managing Custom Buildpacks
- Using Docker in Cloud Foundry
- Managing Domains and Routes
- Creating and Managing Users with the cf CLI
- Creating and Managing Users with the UAA CLI (UAAC)
- Creating and Modifying Quota Plans
- Getting Started with the Notifications Service
- Administering Container-to-Container Networking
- Managing Isolation Segments
- Routing for Isolation Segments
- Feature Flags

For Operators Deploying Cloud Foundry

- Stopping and Starting Virtual Machines
- Managing Diego Cell Limits During Upgrade
- Bulletin Board System (BBS) Data Store Encryption
- Configuring Trusted System Certificates for Applications
- Setting a Maximum Number of Started Containers
- Enabling IPv6 for Hosted Applications
- Securing Traffic into Cloud Foundry
- Enabling TCP Routing
- Troubleshooting TCP Routing
- Supporting WebSockets
- Configuring Load Balancer Healthchecks for Cloud Foundry
- Enabling Zipkin Tracing
- Troubleshooting Slow Requests in Cloud Foundry
- Router Idle Keepalive Connections

Pivotal

Managing Custom Buildpacks

Page last updated:

This topic describes how an admin can manage additional buildpacks in Cloud Foundry using the Cloud Foundry Command Line Interface tool (cf CLI). If your application uses a language or framework that the Cloud Foundry system buildpacks do not support, you can take one of the following actions:

- Write your own buildpack
- Customize an existing buildpack
- Use a Cloud Foundry Community Buildpack
- Use a Heroku Third-Party Buildpack

Add a Buildpack

> **Note**: You must be a Cloud Foundry admin user to run the commands discussed in this section.

To add a buildpack, run the `cf create-buildpack BUILDPACK PATH POSITION [--enable|--disable]` command. The arguments to cf create-buildpack specify the following:

- **BUILDPACK** specifies the buildpack name.
- **PATH** specifies the location of the buildpack. PATH can point to a zip file, the URL of a zip file, or a local directory.
- **POSITION** specifies where to place the buildpack in the detection priority list. For more information, see the Buildpack Detection topic.
- **enable** or **disable** specifies whether to allow apps to be pushed with the buildpack. This argument is optional, and defaults to enable. When a buildpack is disabled, app developers cannot push apps using that buildpack.

To confirm that you have successfully added a buildpack, run `cf buildpacks`.

The following example shows the output from running the `cf buildpacks` command after an administrator adds a Python buildpack:

```
$ cf buildpacks
Getting buildpacks...

buildpack        position  enabled  locked  filename
ruby_buildpack    1         true     false   buildpack_ruby_v46-245-g2fc4ad8.zip
nodejs_buildpack  2         true     false   buildpack_nodejs_v8-177-g2b0a5cf.zip
java_buildpack    3         true     false   buildpack_java_v2.1.zip
python_buildpack  4         true     false   buildpack_python_v2.7.6.zip
```

Rename a Buildpack

To rename a builpack, run `cf rename-buildpack BUILDPACK-NAME NEW-BUILDPACK-NAME`. Replace `BUILDPACK-NAME` with the original buildpack name, and `NEW-BUILDPACK-NAME` with the new buildpack name.

For more information about renaming buildpack, see the cf CLI documentation.

Update a Buildpack

```
$ cf update-buildpack BUILDPACK [-p PATH] [-i POSITION] [--enable|--disable] [--lock|--unlock]
```

For more information about updating buildpacks, see the cf CLI documentation.

Delete a Buildpack

```
$ cf delete-buildpack BUILDPACK [-f]
```

For more information about deleting buildpacks, see the cf CLI documentation.

Lock and Unlock a Buildpack

Every new version of Cloud Foundry includes an updated buildpack. By default, your deployment applies the most recent buildpack when you upgrade. In some cases, however, you may want to preserve an existing buildpack, rather than upgrade to the latest version. For example, if an app you deploy depends on a specific component in Buildpack A that is not available in Buildpack B, you may want to continue using Buildpack A.

The `--lock` flag lets you continue to use your existing buildpack even after you upgrade. Locked buildpacks are not updated when PCF updates. You must manually unlock them to update them.

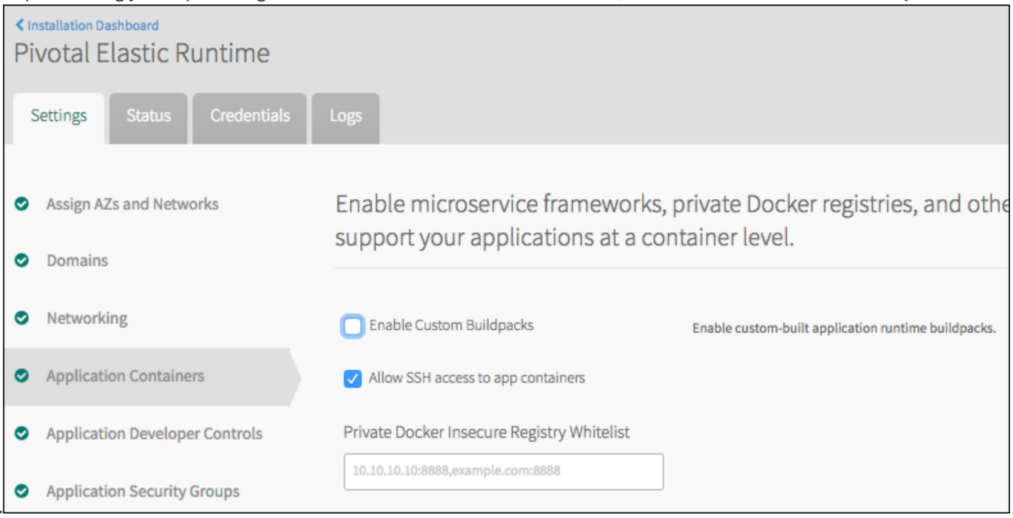
If you elect to use the `--unlock` flag, your deployment will apply the most recent buildpack when you upgrade PCF.

```
cf update-buildpack BUILDPACK [-p PATH] [-i POSITION] [--enable|--disable] [--lock|--
unlock]
```

This feature is also available via API. For more information, see the API documentation.

Disabling Custom Buildpacks

You can disable custom buildpacks using your Ops Manager PAS tile. From the **Cloud Controller** tab, check the **Disable Custom Buildpacks** checkbox, as shown in the image below.



By default, the cf CLI gives developers the option of using a custom buildpack when they deploy apps to PAS. To do so, they use the `-b` option to provide a custom buildpack URL with the `cf push` command. The **Disable Custom Buildpacks** checkbox prevents the `-b` option from being used with external buildpack URLs.

For more information about custom buildpacks, refer to the buildpacks section of the PCF documentation.

Using Docker in Cloud Foundry

Page last updated:

This topic describes how Cloud Foundry (CF) operators can enable CF developers to run their apps in Docker containers, and explains how Docker works in Cloud Foundry.

For information about Diego, the Cloud Foundry component that manages application containers, see the Diego Architecture topic. For information about how CF developers push apps with Docker images, see Deploy an App with Docker.

Enable Docker

By default, apps deployed with the `cf push` command run in standard Cloud Foundry Linux containers. With Docker support enabled, Cloud Foundry can also deploy and manage apps running in Docker containers.

To deploy apps to Docker, developers run `cf push` with the `--docker-image` option and the location of a Docker image to create the containers from. See the Push a Docker Image topic for information about how CF developers push apps with Docker images.

To enable Docker support on a CF deployment, an operator must do the following:

- Enable the `diego_docker` feature flag.
- Configure access to any Docker registries that developers want to use images from.

Enable and Disable the diego_docker Feature Flag

The `diego_docker` feature flag governs whether a CF deployment supports Docker containers.

To enable Docker support, run:

```
$ cf enable-feature-flag diego_docker
```

To disable Docker support, run:

```
$ cf disable-feature-flag diego_docker
```

Note: Disabling the `diego_docker` feature flag stops all Docker-based apps in your deployment within a few convergence cycles, on the order of a minute.

Configure Docker Registry Access

To support Docker, Pivotal Cloud Foundry needs the ability to access Docker registries using either a Certificate Authority or an IP address whitelist. The Using Docker Registries topic explains how to configure this access.

Docker Image Contents

A Docker image consists of a collection of layers. Each layer consists of one or both of the following:

- Raw bits to download and mount. These bits form the file system.
- Metadata that describes commands, users, and environment for the layer. This metadata includes the `ENTRYPOINT` and `CMD` directives, and is specified in the Dockerfile.

How Garden-runC Creates Containers

Diego currently uses Garden-runC to construct Linux containers.

Both Docker and Garden-runC use libraries from the Open Container Initiative (OCI) to build Linux containers. After creation, these containers use name space isolation, or *namespaces*, and control groups, or *cgroups*, to isolate processes in containers and limit resource usage. These are common kernel resource isolation features used by all Linux containers.

Note: PAS versions v1.8.8 and above use Garden-runC instead of Garden-Linux.

Before Garden-runC creates a Linux container, it creates a file system that is mounted as the root file system of the container. Garden-runC supports mounting Docker images as the root file systems for the containers it creates.

When creating a container, both Docker and Garden-runC perform the following actions:

- Fetch and cache the individual layers associated with a Docker image
- Combine and mount the layers as the root file system

These actions produce a container whose contents exactly match the contents of the associated Docker image.

Earlier versions of Diego used Garden-Linux. For more information, see the Garden topic.

How Diego Runs and Monitors Processes

After Garden-runC creates a container, Diego runs and monitors the processes inside of it.

To determine which processes to run, the Cloud Controller fetches and stores the metadata associated with the Docker image. The Cloud Controller uses this metadata to perform the following actions:

- Runs the start command as the user specified in the Docker image
- Instructs Diego and the Gorouter to route traffic to the lowest-numbered port exposed in the Docker image, or port 8080 if the Dockerfile does not explicitly expose a listen port.

Note: When launching an application on Diego, the Cloud Controller honors any user-specified overrides such as a custom start command or custom environment variables.

Docker Security Concerns in a Multi-Tenant Environment

The attack surface area for a Docker-based container running on Diego remains somewhat higher than that of a buildpack application because Docker allows users to fully specify the contents of their root file systems. A buildpack application runs on a trusted root filesystem.

Garden-runC provides features that allow the platform to run Docker images more securely in a multi-tenant context. In particular, Cloud Foundry uses the `user-namespacing` feature found on modern Linux kernels to ensure that users cannot gain escalated privileges on the host even if they escalate privileges within a container.

The Cloud Controller always runs Docker containers on Diego with user namespaces enabled. This security restriction prevents certain features, such as the ability to mount FuseFS devices, from working in Docker containers. Docker applications can use fuse mounts through volume services, but they cannot directly mount fuse devices from within the container.

To mitigate security concerns, Cloud Foundry recommends that you run only trusted Docker containers on the platform. By default, the Cloud Controller does not allow Docker-based applications to run on the platform.

Pivotal

Managing Domains and Routes

Page last updated:

If you are an administrator, you can manage custom shared domains and wildcard routes.

For additional information about managing routes and domains, refer to the following topic:

- Routes and Domains

Creating a Shared Custom Domain

You can use a registered domain of your own and associate it with all organizations in your account. Use the cf create-shared-domain command to create a shared custom domain available to all organizations in your account.

For example:

```
$ cf create-shared-domain shared-domain.example.com
```

Deleting a Shared Custom Domain

Use the cf delete-shared-domain command to delete a shared domain:

```
$ cf delete-shared-domain shared-domain.example.com
```

Note: Deleting a shared domain removes all associated routes, making any application with this domain unreachable.

Creating a Wildcard Route

Use the cf create-route command with a **wildcard route** by specifying the host as `*`. The star operator, `*`, signals a wildcard route to match any URL that uses your domain, regardless of the host.

Note: You must surround the `*` with quotation marks when referencing it using the Cloud Foundry Command Line Interface (cf CLI).

For example, the following command created the wildcard route “*.example.org” in the “development” space:

```
$ cf create-route development example.org -n "*"
```

Creating and Managing Users with the cf CLI

Using the Cloud Foundry Command Line Interface (cf CLI), administrators, Org Managers, and Space Managers can manage users. Cloud Foundry uses role-based access control, with each role granting permissions in either an organization or an application space.

For more information, see Organizations, Spaces, Roles, and Permissions.

Understanding Roles

To manage all users, organizations, and roles with the cf CLI, log in with your admin credentials. In Pivotal Operations Manager, refer to **PAS > Credentials** for the admin name and password.

If the feature flag `set_roles_by_username` is enabled, Org Managers can assign org roles to existing users in their org and Space Managers can assign space roles to existing users in their space. For more information about using feature flags, see the Feature Flags topic.

Creating and Deleting Users

FUNCTION	COMMAND	EXAMPLE
Create a new user	cf create-user USERNAME PASSWORD	`cf create-user Alice pa55w0rd`
Create a new user, specifying LDAP as an external identity provider	cf create-user USERNAME –origin ORIGIN	`cf create-user Aayah ldap`
Create a new user, specifying SAML or OpenID Connect as an external identity provider	cf create-user USERNAME –origin ORIGIN	`cf create-user Aiko provider-alias`
Delete a user	cf delete-user USERNAME	`cf delete-user Alice`

Creating Administrator Accounts

To create a new administrator account, use the UAA CLI.

Note: The cf CLI cannot create new administrator accounts.

Org and App Space Roles

A user can have one or more roles. The combination of these roles defines the user's overall permissions in the org and within specific app spaces in that org.

Org Roles

Valid org roles are OrgManager, BillingManager, and OrgAuditor.

FUNCTION	COMMAND	EXAMPLE
View the organizations belonging to an account	cf orgs	`cf orgs`
View all users in an organization by role	cf org-users ORGANIZATION-NAME	`cf org-users my-example-org`
Assign an org role to a user	cf set-org-role USERNAME ORGANIZATION-NAME ROLE	`cf set-org-role Alice my-example-org OrgManager`
Remove an org role from a user	cf unset-org-role USERNAME ORGANIZATION-NAME ROLE	`cf unset-org-role Alice my-example-org OrgManager`

App Space Roles

Each app space role applies to a specific app space.

Valid app space roles are SpaceManager, SpaceDeveloper, and SpaceAuditor.

FUNCTION	COMMAND	EXAMPLE
View the spaces in an org	cf spaces	`cf spaces`
View all users in a space by role	cf space-users ORGANIZATION-NAME SPACE-NAME	`cf space-users my-example-org development`
Assign a space role to a user	cf set-space-role USERNAME ORGANIZATION-NAME SPACE-NAME ROLE	`cf set-space-role Alice my-example-org development SpaceAuditor`
Remove a space role from a user	cf unset-space-role USERNAME ORGANIZATION-NAME SPACE-NAME ROLE	`cf unset-space-role Alice my-example-org development SpaceAuditor`

Creating and Managing Users with the UAA CLI (UAAC)

Page last updated:

Using the UAA Command Line Interface (UAAC), an administrator can create users in the User Account and Authentication (UAA) server.

Note: The UAAC only creates users in UAA, and does not assign roles in the Cloud Controller database (CCDB). In general, administrators create users using the Cloud Foundry Command Line Interface (cf CLI). The cf CLI both creates user records in the UAA and associates them with org and space roles in the CCDB. Before administrators can assign roles to the user, the user must log in through Apps Manager or the cf CLI for the user record to populate the CCDB. Review the Creating and Managing Users with the cf CLI topic for more information.

For additional details and information, refer to the following topics:

- UAA Overview
- UAA Sysadmin Guide
- Other UAA Documentation

Note: UAAC requires Ruby v2.3.1 or later. If you have an earlier version of Ruby installed, install v2.3.1 or later before using the UAAC.

For more information about which roles can perform various operations in Cloud Foundry, see the Roles and Permissions topic.

Create an Admin User

1. Install the UAA CLI, `uaac`.

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

2. Use the `uaac target uaa.YOUR-DOMAIN` command to target your UAA server.

```
$ uaac target uaa.example.com
```

3. Record the **uaa:admin:client_secret** from your deployment manifest.

4. Run `uaac token client get admin -s ADMIN-CLIENT-PASSWORD` to authenticate and obtain an access token for the admin client from the UAA server. Replace `ADMIN-CLIENT-PASSWORD` with your admin password. UAAC stores the token in `~/.uaac.yml`.

```
$ uaac token client get admin -s MyAdminPassword
```

5. Use the `uaac contexts` command to display the users and applications authorized by the UAA server, and the permissions granted to each user and application.

```
$ uaac contexts

[1]*[admin]
  client_id: admin
    access_token: aBcdEfg0hIJKlm123.e
    token_type: bearer
    expires_in: 43200
    scope: uaa.admin clients.secret scim.read
    jti: 91b3-abcd1233
```

6. In the output from `uaac contexts`, search in the `scope` section of the `client_id: admin` user for **scim.write**. The value **scim.write** represents sufficient permissions to create accounts.

7. If the admin user lacks permissions to create accounts, add the permissions by following these steps:
   - Run `uaac client update admin --authorities "EXISTING-PERMISSIONS scim.write"` to add the necessary permissions to the admin user account on the UAA server. Replace EXISTING-PERMISSIONS with the current contents of the `scope` section from `uaac contexts`.
   - Run `uaac token delete` to delete the local token.
   - Run `uaac token client get admin` to obtain an updated access token from the UAA server.

```
$ uaac contexts

[1]*[admin]
  client_id: admin
  . . .
  scope: uaa.admin clients.secret scim.read
  . . .

$ uaac client update admin --authorities "`uaac client get admin | \
  awk '/:/{e=0}/authorities:/{e=1;if(e==1){$1="";print}}'` scim.write"

$ uaac token delete
$ uaac token client get admin
```

8. Run the following command to create an admin user: `uaac user add NEW-ADMIN-USERNAME -p NEW-ADMIN-PASSWORD --emails NEW-ADMIN-EMAIL` Replace `NEW-ADMIN-USERNAME`, `NEW-ADMIN-PASSWORD`, and `NEW-ADMIN-EMAIL` with appropriate information.

```
$ uaac user add Adam -p newAdminSecretPassword --emails newadmin@example.com
```

9. Run `uaac member add GROUP NEW-ADMIN-USERNAME` to add the new admin to the groups `cloud_controller.admin`, `uaa.admin`, `scim.read`, and `scim.write`.

```
$ uaac member add cloud_controller.admin Adam
$ uaac member add uaa.admin Adam
$ uaac member add scim.read Adam
$ uaac member add scim.write Adam
```

Create an Admin Read-Only User

The admin read-only account can view but not modify all Cloud Controller API resources.

If you want to create an admin read-only user account, then perform the following steps:

1. Obtain the credentials of an admin client created using UAAC as above, or refer to the `uaa: scim` section of your deployment manifest for the user name and password of an admin user.

2. Run `uaac token client get admin -s ADMIN-CLIENT-PASSWORD` to authenticate and obtain an access token for the admin client from the UAA server. Replace `ADMIN-CLIENT-PASSWORD` with your admin password. UAAC stores the token in `~/.uaac.yml`.

```
$ uaac token client get admin -s MyAdminPassword
```

3. Run the following command to create an admin read-only user: `uaac user add NEW-USERNAME -p NEW-PASSWORD --emails NEW-EMAIL` Replace `NEW-USERNAME`, `NEW-PASSWORD`, and `NEW-EMAIL` with appropriate information.

```
$ uaac user add Bob -p SecretPassword --emails bob@example.com
```

4. Run `uaac member add GROUP NEW-USERNAME` to add the new admin read-only account to the groups `cloud_controller.admin_read_only` and `scim.read`.

```
$ uaac member add cloud_controller.admin_read_only Bob
$ uaac member add scim.read Bob
```

Create a Global Auditor

The global auditor account has read-only access to all Cloud Controller API resources but cannot access secret data such as environment variables.

Perform the following steps to create a global auditor account.

1. Obtain the credentials of an admin client created using UAAC as above, or refer to the `uaa: scim` section of your deployment manifest for the user name and password of an admin user.

2. Run `uaac token client get admin -s ADMIN-CLIENT-PASSWORD` to authenticate and obtain an access token for the admin client from the UAA server. Replace `ADMIN-CLIENT-PASSWORD` with your admin password. UAAC stores the token in `~/.uaac.yml`.

```
$ uaac token client get admin -s MyAdminPassword
```

Pivotal

3. Run `uaac user add NEW-USERNAME -p NEW-PASSWORD --emails NEW-EMAIL` to create a global auditor user. Replace `NEW-USERNAME`, `NEW-PASSWORD`, and `NEW-EMAIL` with appropriate information.

```
$ uaac user add Alice -p SecretPassword --emails alice@example.com
```

4. Ensure that the `cloud_controller.global_auditor` group exists.

```
$ uaac group add cloud_controller.global_auditor
```

5. Run `uaac member add GROUP NEW-USERNAME` to add the new global auditor account to the `cloud_controller.global_auditor` group.

```
$ uaac member add cloud_controller.global_auditor Alice
```

Grant Admin Permissions to an External Group (SAML or LDAP)

Follow the steps below to grant all users under an external group admin permissions.

1. Obtain the credentials of an admin client created using UAAC as above, or refer to the `uaa: scim` section of your deployment manifest for the user name and password of an admin user.

2. Run `uaac token client get admin -s ADMIN-CLIENT-PASSWORD` to authenticate and obtain an access token for the admin client from the UAA server. Replace `ADMIN-CLIENT-PASSWORD` with your admin password. UAAC stores the token in `~/.uaac.yml`.

```
$ uaac token client get admin -s MyAdminPassword
```

3. Run the commands below to grant all users under the mapped LDAP Group admin permissions. Replace `GROUP-DISTINGUISHED-NAME` with an appropriate group name.
   - `uaac group map --name scim.read "GROUP-DISTINGUISHED-NAME"`
   - `uaac group map --name cloud_controller.admin "GROUP-DISTINGUISHED-NAME"`

4. Retrieve the name of your SAML provider by navigating to the PAS tile on the Ops Manager Installation Dashboard, clicking**Authentication and Enterprise SSO**, and recording the value under **Provider Name**. For more information about configuring PCF for a SAML identity provider, see the Configuring Authentication and Enterprise SSO for PAS topic.

5. Run the commands below to grant all users under the mapped SAML group admin permissions. Replace `GROUP-NAME` with the group name, and `SAML-PROVIDER-NAME` with the name of your SAML provider.
   - `uaac group map --name scim.read "GROUP-NAME" --origin SAML-PROVIDER-NAME`
   - `uaac group map --name cloud_controller.admin "GROUP-NAME" --origin SAML-PROVIDER-NAME`

Create Users

1. Obtain the credentials of an admin client created using UAAC as above, or refer to the `uaa: scim` section of your deployment manifest for the user name and password of an admin user.

2. Run `cf login -u NEW-ADMIN-USERNAME -p NEW-ADMIN-PASSWORD` to log in.

```
$ cf login -u Adam -p newAdminSecretPassword
```

3. Run `cf create-user NEW-USER-NAME NEW-USER-PASSWORD` to create a new user.

```
$ cf create-user Charlie aNewPassword
```

Change Passwords

1. Obtain the credentials of an admin client created using UAAC as above, or refer to the `uaa: scim` section of your deployment manifest for the user name and password of an admin user.

2. Run `uaac token client get admin -s ADMIN-CLIENT-PASSWORD` to authenticate and obtain an access token for the admin client from the UAA server.

Replace `ADMIN-CLIENT-PASSWORD` with your admin password. UAAC stores the token in `~/.uaac.yml`.

```
$ uaac token client get admin -s MyAdminPassword
```

3. Run `uaac contexts` to display the users and applications authorized by the UAA server, and the permissions granted to each user and application.

```
$ uaac contexts

[1]*[admin]
  client_id: admin
    access_token: aBcdEfg0hIJKlm123.e
    token_type: bearer
    expires_in: 43200
    scope: uaa.admin clients.secret password.read
    jti: 91b3-abcd1233
```

4. In the output from `uaac contexts`, search in the `scope` section of the `client_id: admin` user for **password.write**. The value **password.write** represents sufficient permissions to change passwords.

5. If the admin user lacks permissions to change passwords, add the permissions by following these steps:
   - Run `uaac client update admin --scope "EXISTING-PERMISSIONS password.write"` to add the necessary permissions to the admin user account on the UAA server. Replace EXISTING-PERMISSIONS with the current contents of the `scope` section from `uaac contexts`.
   - Run `uaac token delete` to delete the local token.
   - Run `uaac token client get admin` to obtain an updated access token from the UAA server.

```
$ uaac contexts

[1]*[admin]
  client_id: admin
  . . .
  scope: uaa.admin clients.secret password.read
  . . .

$ uaac client update admin --scope "`uaac client get admin | \
  awk '/:/{e=0}/authorities:/{e=1;if(e==1){$1="";print}}'` password.write"

$ uaac token delete
$ uaac token client get admin
```

6. Run `uaac password set USER-NAME -p TEMP-PASSWORD` to change an existing user password to a temporary password.

```
$ uaac password set Charlie -p ThisIsATempPassword
```

7. Provide the `TEMP-PASSWORD` to the user. Have the user use `cf target api.YOUR-DOMAIN`, `cf login -u USER-NAME -p TEMP-PASSWORD`, and `cf passwd` to change the temporary password. See the Configuring UAA Password Policy topic to configure the password policy.

```
$ cf target api.example.com
$ cf login -u Charlie -p ThisIsATempPassword
$ cf passwd


Current Password>ThisIsATempPassword


New Password>*******


Verify Password>*******
Changing password...
```

Retrieve User Email Addresses

Some Cloud Foundry components, like Cloud Controller, only use GUIDs for user identification. You can use the UAA to retrieve the emails of your Cloud Foundry instance users either as a list or, for a specific user, with that user's GUID.

Follow the steps below to retrieve user email addresses:

1. Run `uaac target uaa.YOUR-DOMAIN` to target your UAA server.


```
$ uaac target uaa.example.com
```

2. Record the **uaa:admin:client_secret** from your deployment manifest.

3. Run `uaac token client get admin -s ADMIN-CLIENT-PASSWORD` to authenticate and obtain an access token for the admin client from the UAA server. Replace `ADMIN-CLIENT-PASSWORD` with your admin password. UAAC stores the token in `~/.uaac.yml`.

```
$ uaac token client get admin -s MyAdminPassword
```

4. Run `uaac contexts` to display the users and applications authorized by the UAA server, and the permissions granted to each user and application.

```
$ uaac contexts

[1]*[admin]
  client_id: admin
    access_token: aBcdEfg0hIJKlm123.e
    token_type: bearer
    expires_in: 43200
    scope: uaa.admin clients.secret
    jti: 91b3-abcd1233
```

5. In the output from `uaac contexts`, search in the `scope` section of the `client_id: admin` user for **scim.read**. The value **scim.read** represents sufficient permissions to query the UAA server for user information.

6. If the admin user lacks permissions to query the UAA server for user information, add the permissions by following these steps:
   - Run `uaac client update admin --authorities "EXISTING-PERMISSIONS scim.write"` to add the necessary permissions to the admin user account on the UAA server. Replace EXISTING-PERMISSIONS with the current contents of the `scope` section from `uaac contexts`.
   - Run `uaac token delete` to delete the local token.
   - Run `uaac token client get admin` to obtain an updated access token from the UAA server.

```
$ uaac contexts

[1]*[admin]
  client_id: admin
  . . .
  scope: uaa.admin clients.secret
  . . .

$ uaac client update admin --authorities "uaa.admin clients.secret scim.read"

$ uaac token delete
$ uaac token client get admin
```

7. Run `uaac users` to list your Cloud Foundry instance users. By default, the `uaac users` command returns information about each user account including GUID, name, permission groups, activity status, and metadata. Use the `--attributes emails` or `-a emails` flag to limit the output of `uaac users` to email addresses.

```
$ uaac users --attributes emails

resources:
  emails:
value: user1@example.com
  emails:
value: user2@example.com
  emails:
value: user3@example.com
```

8. Run `uaac users "id eq GUID" --attributes emails` with the GUID of a specific user to retrieve that user's email address.

```
$ uaac users "id eq 'aabbcc11-22a5-87-8056-beaf84'" --attributes emails

resources:
  emails:
value: user1@example.com
```

Creating and Modifying Quota Plans

Page last updated:

Quota plans are named sets of memory, service, and instance usage quotas. For example, one quota plan might allow up to 10 services, 10 routes, and 2 GB of RAM, while another might offer 100 services, 100 routes, and 10 GB of RAM. Quota plans have user-friendly names, but are referenced in Cloud Foundry (CF) internal systems by unique GUIDs.

Quota plans are not directly associated with user accounts. Instead, every org has a list of available quota plans, and the account admin assigns a specific quota plan from the list to the org. Everyone in the org shares the quotas described by the plan. There is no limit to the number of defined quota plans an account can have, but only one plan can be assigned at a time.

You must set a quota plan for an org, but you can choose whether to set a space quota.

Org Quota Plan Attributes

Name	Description	Valid Values	Example Value
name	The name you use to identify the plan	A sequence of letters, digits, and underscore characters. Quota plan names within an account must be unique.	silver_quota
memory_limit	Maximum memory usage allowed	An integer and a unit of measurement like M, MB, G, or GB	2048M
app_instance_limit	Maximum app instances allowed. Stopped apps do not count toward this instance limit. Crashed apps count toward the limit because their desired state is `starting`.	An integer	25
non_basic_services_allowed	Determines whether users can provision instances of non-free service plans. Does not control plan visibility. When false, non-free service plans may be visible in the marketplace but instances can not be provisioned.	`true` or `false`	true
total_routes	Maximum routes allowed	An integer	500
total_reserved_route_ports	Maximum routes with reserved ports	An integer not greater than total_routes	60
total_services	Maximum services allowed	An integer	25
trial_db_allowed	Legacy Field. Value can be ignored.	`true` or `false`	true

Default Quota Plan for an Org

Cloud Foundry installs with a quota plan named `default` with the following values:

- Memory Limit: `10240 MB`
- Total Routes: `1000`
- Total Services: `100`
- Non-basic Services Allowed: `True`
- Trial DB Allowed: `True`

Create a New Quota Plan for an Org

> **Note:** The org manager sets and manages quotas. See the Orgs, Spaces, Roles, and Permissions topic for more information.

You must set an org quota. You can create a new quota plan for org with `cf create-quota`.

Use cf create-quota

In a terminal window, run the following command. Replace the placeholder attributes with the values for this quota plan:

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

This command accepts the following flags:

- `-m` : Total amount of memory
- `-i` : Maximum amount of memory an application instance can have (`-1` represents an unlimited amount)
- `-r` : Total number of routes
- `-s` : Total number of service instances
- `-allow-paid-service-plans` : Can provision instances of paid service plans

Example:

```
$ cf create-quota small -m 2048M -i 1024M -r 10 -s 10 --allow-paid-service-plans
```

Modify an Existing Quota Plan for an Org

Use cf update-quota

1. Run `cf quotas` to find the names of all quota definitions available to your org. Record the name of the quota plan to be modified.

```
$ cf quotas
Getting quotas as admin@example.com...
OK

name     total memory limit   instance memory limit   routes    service instances   paid service plans
free       0            0               1000      0             disallowed
paid       10G            0               1000       -1            allowed
small       2G            0                 10       10            allowed
trial      2G            0               1000      10            disallowed
```

2. Run the following command, replacing QUOTA with the name of your quota.
`cf update-quota QUOTA [-i INSTANCE-MEMORY] [-m MEMORY] [-n NEW-NAME] [-r ROUTES] [-s SERVICE-INSTANCES] [--allow-paid-service-plans | --disallow-paid-service-plans]`

This command accepts the following flags:
- `-i` : Maximum amount of memory an application instance can have (`-1` represents an unlimited amount)
- `-m` : Total amount of memory a space can have
- `-n` : New name
- `-r` : Total number of routes
- `-s` : Total number of service instances
- `--allow-paid-service-plans` : Can provision instances of paid service plans
- `--disallow-paid-service-plans` : Can not provision instances of paid service plans

Example:

```
$ cf update-quota small -i 2048M -m 4096M -n medium -r 20 -s 20 --allow-paid-service-plans
```

Create and Modify Quota Plans for a Space

For each org, Org Managers create and modify quota plans for spaces in the org. If an Org Manager allocates a space quota, CF verifies that resources do not exceed the allocated space limit. For example, when a Space Developer deploys an application, CF first checks the memory allocation at the space level, then at the org level.

Perform the following procedures to create and modify quota plans for individual spaces within an org.

Create a New Quota Plan for a Space

In a terminal window, run the following command to create a quota for a space. Replace the placeholder attributes with the values for this quota plan:

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

Example:

```
$ cf create-space-quota big -i 1024M -m 4096M -r 20 -s 20 --allow-paid-service-plans
```

Modify a Quota Plan for a Space

Run `cf space-quotas` to find the names of all space quota available to your org. Record the name of the quota plan to be modified.

```
$ cf space-quotas
Getting quotas as admin@example.com...
OK

name     total memory limit   instance memory limit   routes   service instances   paid service plans
big      2G                   unlimited               0        10                  allowed
trial    2G                   0                       0        10                  allowed
```

To modify that quota, use the `update-space-quota` command. Replace the placeholder attributes with the values for this quota plan.

```
cf update-space-quota SPACE-QUOTA-NAME [-i MAX-INSTANCE-MEMORY] [-m MEMORY] [-n NEW-NAME] [-r ROUTES] [-s SERVICES] [--allow-paid-service-plans | --disallow-paid-service-plans]
```

Example:

```
$ cf update-space-quota big -i 20 -m 4096M -n bigger -r 20 -s 20 --allow-paid-service-plans
```

Run cf help

For more information regarding quotas, run `cf help` to view a list and brief description of all cf CLI commands. Scroll to view org and space quotas usage and information.

```
$ cf help
...
ORG ADMIN:
  quotas                       List available usage quotas
  quota                        Show quota info
  set-quota                    Assign a quota to an org

  create-quota                 Define a new resource quota
  delete-quota                 Delete a quota
  update-quota                 Update an existing resource quota

  share-private-domain         Share a private domain with an org
  unshare-private-domain       Unshare a private domain with an org

SPACE ADMIN:
  space-quotas                 List available space resource quotas
  space-quota                  Show space quota info
  create-space-quota           Define a new space resource quota
  update-space-quota           update an existing space quota
  delete-space-quota           Delete a space quota definition and unassign the space quota from all spaces
  set-space-quota              Assign a space quota definition to a space
  unset-space-quota            Unassign a quota from a space
```

Getting Started with the Notifications Service

Page last updated:

This topic describes how to use the Notifications Service, including how to create a client, obtain a token, register notifications, create a custom template, and send notifications to your users.

Prerequisites

- Install Pivotal Application Service.
- You must have `admin` permissions on your Cloud Foundry instance. You also must configure Application Security Groups (ASGs).
- Install the Cloud Foundry Command Line Interface (cf CLI) and User Account and Authorization Server (UAAC) command line tools.

Create a Client and Get a Token

To interact with the Notifications Service, you must create UAA scopes:

1. Use `uaac target uaa.YOUR-DOMAIN` to target your UAA server.

```
$ uaac target uaa.example.com
```

2. Record the **Admin Client Credentials** from the **UAA** row in the PAS **Credentials** tab.

3. Use `uaac token client get admin -s ADMIN-CLIENT-SECRET` to authenticate and obtain an access token for the admin client from the UAA server. UAAC stores the token in `~/.uaac.yml`.

```
$ uaac token client get admin -s MyAdminPassword
```

4. Create a `notifications-admin` client with the required scopes.

```
$ uaac client add notifications-admin --authorized_grant_types client_credentials --authorities \
  notifications.manage,notifications.write,notification_templates.write,notification_templates.read,critical_notifications.write
```

   - `notifications.write`: send a notification. For example, you can send notifications to a user, space, or everyone in the system.
   - `notifications.manage`: update notifications and assign templates for that notification.
   - (Optional) `notification_templates.write`: create a custom template for a notification.
   - (Optional) `notification_templates.read`: check which templates are saved in the database.

5. Log in using your newly created client:

```
$ uaac token client get notifications-admin
```

Note: Stay logged in to this client to follow the examples in this topic.

Register Notifications

Note: To register notifications, you must have the `notifications.manage` scope on the client. To set critical notifications, you must have the `critical_notifications.write` scope.

You must register a notification before sending it. Using the token `notifications-admin` from the previous step, the following example registers two notifications with the following properties:

```
$ uaac curl https://notifications.user.example.com/notifications -X PUT --data '{  "source_name": "Cloud Ops Team",
  "notifications": {
    "system-going-down": {"critical": true, "description": "Cloud going down" },
    "system-up": { "critical": true, "description": "Cloud back up" }
    }
  }'
```

- `source_name` has "Cloud Ops Team" set as the description.
- `system-going-down` and `system-up` are the notifications set.
- `system-going-down` and `system-up` are made `critical`, so no users can unsubscribe from that notification.

Create a Custom Template

Note: To view a list of templates, you must have the `notifications_templates.read` scope. To create a custom template, you must have the `notification_templates.write` scope.

A template is made up of a name, a subject, a text representation of the template you are sending for mail clients that do not support HTML, and an HTML version of the template.

The system provides a default template for all notifications, but you can create a custom template using the following curl command.

```
$ uaac curl https://notifications.user.example.com/templates -X POST --data \
'{"name":"site-maintenance","subject":"Maintenance: {{.Subject}}","text":"The site has gone down for maintenance. More information to follow {{.Text}}","html":"

The site has gone down for maintenance. More information to follow {{.HTML}}"}'
```

Variables that take the form `{{.}}` interpolate data provided in the send step before a notification is sent. Data that you can insert into a template during the send step include `{{.Text}}`, `{{.HTML}}`, and `{{.Subject}}`.

This curl command returns a unique template ID that can be used in subsequent calls to refer to your custom template. The result looks similar to this:

`{"template-id": "E3710280-954B-4147-B7E2-AF5BF62772B5"}`

Check all of your saved templates by running a curl command:

```
$ uaac curl https://notifications.user.example.com/templates -X GET
```

Associate a Custom Template with a Notification

In this example, the `system-going-down` notification belonging to the `notifications-admin` client is associated with the template ID `E3710280-954B-4147-B7E2-AF5BF62772B5`. This is the template ID of the template we created in the previous section.

Associating a template with a notification requires the `notifications.manage` scope.

```
$ uaac curl https://notifications.user.example.com/clients/notifications-admin/notifications/system-going-down/template \
-X PUT --data '{"template": "E3710280-954B-4147-B7E2-AF5BF62772B5"}'
```

Any notification that does not have a custom template applied, such as `system-up`, defaults to a system-provided template.

Administering Container-to-Container Networking

This topic describes how to configure the Container-to-Container Networking feature. For an overview of how Container-to-Container Networking works, see the Understanding Container-to-Container Networking topic.

Container-to-Container Networking enables PCF to generate logs whenever containers communicate or attempt to communicate with each other. See App Traffic Logging for how to manage app traffic logging.

Create Policies for Container-to-Container Networking

This section describes how to create and modify Container-to-Container Networking policies using the Cloud Foundry Command Line Interface (cf CLI).

Note: With the NSX-T integration, container networking policies and ASGs continue to work as normal. Advanced ASG logging is not supported with NSX-T.

Ensure that you are using cf CLI v6.30 or higher:

```
$ cf version
```

For more information about updating the cf CLI, see the Installing the cf CLI topic.

To grant specific users or groups the permissions to configure network policies, use the following UAA scopes:

UAA Scope	Suitable for…	Allows users to create policies…
`network.admin`	operators	for any apps in the CF deployment
`network.write`	space developers	for apps in spaces that they can access

If you are a CF admin, you already have the `network.admin` scope. An admin can also grant the `network.admin` scope to a space developer.

For more information, see Creating and Managing Users with the UAA CLI (UAAC) and Orgs, Spaces, Roles, and Permissions.

To grant all Space Developers permissions to configure network policies, select the **Allow Space Developers to manage network policies** checkbox in the **Application Developer Controls** tab of the PAS tile.

Add a Network Policy

To add a policy that allows direct network traffic from one app to another, run the following command:

```
cf add-network-policy SOURCE_APP --destination-app DESTINATION_APP --protocol (tcp | udp) --port RANGE
```

Replace the placeholders in the above command as follows:

- `SOURCE_APP` is the name of the app that sends traffic.
- `DESTINATION_APP` is the name of the app that will receive traffic.
- `PROTOCOL` is one of the following: `tcp` or `udp`.
- `RANGE` are the ports at which to connect to the destination app. The allowed range is from `1` to `65535`. You can specify a single port, such as `8080`, or a range of ports, such as `8080-8090`.

The following example command allows access from the `frontend` app to the `backend` app over TCP at port 8080:

```
$ cf add-network-policy frontend --destination-app backend --protocol tcp --port 8080
Adding network policy to app frontend in org my-org / space dev as admin...
OK
```

List Policies

You can list all the policies in your space, or just the policies for which a single app is the source:

- To list the all the policies in your space, run `cf network-policies`.

```
$ cf network-policies
```

- To list the policies for an app, run `cf network-policies --source MY-APP`. Replace `MY-APP` with the name of your app.

```
$ cf network-policies --source example-app
```

  The following example command lists policies for the app `frontend`:

```
$ cf network-policies --source frontend
Listing network policies in org my-org / space dev as admin...

source     destination   protocol   ports
frontend   backend       tcp        8080
```

Remove a Network Policy

To remove a policy that allows direct network traffic from an app, run the following command:

`cf remove-network-policy SOURCE_APP --destination-app DESTINATION_APP --protocol PROTOCOL --port RANGE`

Replace the placeholders in the above command to match an existing policy, as follows:

- `SOURCE_APP` is the name of the app that sends traffic.
- `DESTINATION_APP` is the name of the app that receives traffic.
- `PROTOCOL` is either `tcp` or `udp`.
- `PORTS` are the ports connecting the apps. The allowed range is from `1` to `65535`. You can specify a single port, such as `8080`, or a range of ports, such as `8080-8090`.

The following command deletes the policy that allowed the `frontend` app to communicate with the `backend` app over TCP on port 8080:

```
$ cf remove-network-policy frontend --destination-app backend --protocol tcp --port 8080
Removing network policy to app frontend in org my-org / space dev as admin...
OK
```


Managing Isolation Segments

In this topic:

This topic describes how operators can isolate deployment workloads into dedicated resource pools called isolation segments.

Requirements

You must have the v.6.26.0 version or later of the Cloud Foundry Command Line Interface (cf CLI) installed to manage isolation segments.

Target the API endpoint of your deployment with `cf api` and log in with `cf login` before performing the procedures in this topic. For more information, see the Identifying the API Endpoint for your PAS Instance topic.

Overview

To enable isolation segments, an operator must install the PCF Isolation Segment tile by performing the procedures in the Installing PCF Isolation Segment topic. Installing the tile creates a single isolation segment.

After an admin creates a new isolation segment, the admin can then create and manage relationships between the orgs and spaces of a Cloud Foundry deployment and the new isolation segment.

To manage the isolation segment, an operator uses cf CLI commands.

Operators can perform the following operations on isolation segments:

- Create an isolation segment
- List isolation segments
- Display isolation segments enabled for an org
- Show the isolation segment assigned to a space
- Delete an isolation segment
- Enable isolation segments for an org
- Disable isolation segments for an org
- Set the default isolation segment for an org
- Assign an isolation segment to a space
- Reset the isolation segment assignment for a space

Create an Isolation Segment

Before you create an isolation segment in PCF, you must install the PCF Isolation Segment tile by performing the procedures in the Installing PCF Isolation Segment topic.

To register an isolation segment with Cloud Controller, use the cf CLI.

> **Note:** The isolation segment name used in the cf CLI command must match the value specified in the **Segment Name** field of the PCF Isolation Segment tile. If the names do not match, PCF fails to place apps in the isolation segment when apps are started or restarted in the space assigned to the isolation segment.

The following command creates an isolation segment named `my_segment`:

```
$ cf create-isolation-segment my_segment
```

If successful, the command returns an `OK` message:

```
Creating isolation segment my_segment as admin...
OK
```

Retrieve Isolation Segment Information

The `cf isolation-segments`, `cf org`, and `cf space` commands retrieve information about isolation segments. The isolation segments you can see depends on your role, as follows:

- **Admins** see all isolation segments in the system.
- **Other users** only see the isolation segments that their orgs are entitled to.

List Isolation Segments

The following request returns a list of the isolation segments that are available to you:

```
$ cf isolation-segments
```

For example, the command returns results similar to the following:

```
Getting isolation segments as admin...
OK

name          orgs
my_segment    org1, org2
```

Display Isolation Segments Enabled for an Org

An admin can entitle an org to multiple isolation segments.

Run `cf org ORG-NAME` command to view the isolation segments that are available to an org. Replace `ORG-NAME` with the name of your org.

For example:

```
$ cf org my-org
```

The command returns results similar to the following:

```
Getting info for org my-org as user@example.com...

name:                 my-org
domains:              example.com, apps.example.com
quota:                paid
spaces:               development, production, sample-apps, staging
isolation segments:   my_segment, my_other_segment
```

Show the Isolation Segment Assigned to a Space

Only one isolation segment can be assigned to a space.

Run `cf space SPACE-NAME` to view the isolation segment assigned to a space. Replace `SPACE-NAME` with the name of the space.

For example:

```
$ cf space staging
```

The command returns results similar to the following:

```
name:              staging
org:               my-org
apps:
services:
isolation segment:   my_segment
space quota:
security groups:     dns, p-mysql, p.mysql, pcf-redis, public_networks, rabbitmq, ssh-logging
```

Delete an Isolation Segment

Note: An isolation segment with deployed apps cannot be deleted.

Only admins can delete isolation segments.

Run `cf delete-isolation-segment SEGMENT-NAME` to delete an isolation segment. Replace `SEGMENT-NAME` with the name of the isolation segment. If successful, the command returns an `OK` message.

For example:

```
$ cf delete-isolation-segment my_segment
Deleting isolation segment my_segment as admin...
OK
```

Manage Isolation Segment Relationships

The commands listed in the sections below manage the relationships between isolation segments, orgs, and spaces.

Enable an Org to Use Isolation Segments

Only admins can enable orgs to use isolation segments. Run `cf enable-org-isolation ORG-NAME SEGMENT-NAME` to enable the use of an isolation segment. Replace `ORG-NAME` with the name of your org, and `SEGMENT-NAME` with the name of the isolation segment.

For example:

```
$ cf enable-org-isolation org2 my_segment
```

If an org is entitled to use only one isolation segment, that isolation segment does not automatically become the default isolation segment for the org. You must explicitly set the default isolation segment of an org.

Disable an Org from Using Isolation Segments

Note: You cannot disable an org from using an isolation segment if a space within that org is assigned to the isolation segment. Additionally, you cannot disable an org from using an isolation segment if the isolation segment is configured as the default for that org.

Run `cf disable-org-isolation ORG-NAME SEGMENT-NAME` to disable an org from using an isolation segment. Replace `ORG-NAME` with the name of your org, and `SEGMENT-NAME` with the name of the isolation segment.

For example:

```
$ cf disable-org-isolation org1 my_segment
```

If successful, the command returns an `OK` message:

```
Removing entitlement to isolation segment my_segment from org org1 as admin...
OK
```

Set the Default Isolation Segment for an Org

This section requires [cf CLI v6.29.0](#) or later.

Only admins and org managers can set the default isolation segment for an org.

When an org has a default isolation segment, apps in its spaces belong to the default isolation segment unless you assign them to another isolation segment. You must restart running applications to move them into the default isolation segment.

Run `cf set-org-default-isolation-segment ORG-NAME SEGMENT-NAME` to set the default isolation segment for an org. Replace `ORG-NAME` with the name of your org, and `SEGMENT-NAME` with the name of the isolation segment.

For example:

```
$ cf set-org-default-isolation-segment org1 my_segment
Setting isolation segment my_segment to default on org org1 as admin...
OK
```

To display the default isolation segment for an org, use the `cf org` command.

Assign an Isolation Segment to a Space

Admins and org managers can assign an isolation segment to a space. Apps in that space start in the specified isolation segment.

To assign an isolation segment to a space, you must first enable the space's org to use the isolation segment. See [Enable an Org to Use Isolation Segments](#)

Run `cf set-space-isolation-segment SPACE-NAME SEGMENT-NAME` to assign an isolation segment to a space. Replace `SPACE-NAME` with the name of the space, and `SEGMENT-NAME` with the name of the isolation segment.

For example:

```
$ cf set-space-isolation-segment space1 my_segment
```

Reset the Isolation Segment Assignment for a Space

Admins can reset the isolation segment assigned to a space to use the org's default isolation segment.

Run `cf reset-space-isolation-segment SPACE-NAME` to the assign the default isolation segment for an org to a space. Replace `SPACE-NAME` with the name of the space.

For example:

```
$ cf reset-space-isolation-segment space1
```

Routing for Isolation Segments

Page last updated:

This topic describes how operators can configure and manage routing for isolation segments. Operators can deploy an additional set of routers for each isolation segment to handle requests for applications within the segment. This topic includes the following sections:

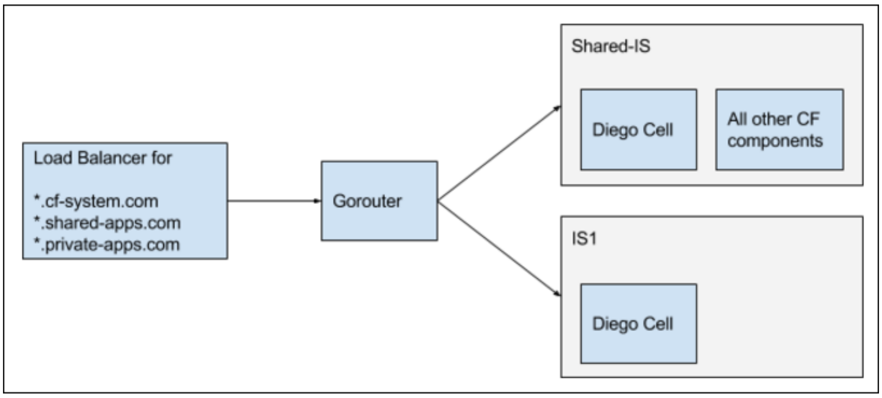
For more information about how isolation segments work, see the Isolation Segments section of the *Understanding Cloud Foundry Security* topic. For more information about creating isolation segments, see the Installing PCF Isolation Segment topic.

Note: The instructions in this topic assume you are using Google Cloud Platform (GCP). The procedures may differ on other IaaSes, but the concepts should be transferable.

Overview

Isolation segments isolate the compute resources for one group of applications from another. However, these applications still share the same network resources. Requests for applications on all isolation segments, as well as for system components, transit the same load balancers and Cloud Foundry routers.

The illustration below shows isolation segments sharing the same network resources.



Operators who want to prevent all isolation segments and system components from using the same network resources can deploy an additional set of routers for each isolation segment. Use cases include:

- Requests for applications in an isolation segment must not share networking resources with requests for other applications.
- The Cloud Foundry management plane should only be accessible from a private network. As multiple IaaS load balancers cannot typically share the same pool of backends, such as Cloud Foundry routers, each load balancer requires an additional deployment of routers.

Step 1: Create Networks

Create a network or subnet for each isolation segment on your infrastructure

As an example, an operator who wants one shared isolation segment and two private segments could create one network named `sample-network` with three subnets named `sample-subnet-shared`, `sample-subnet-is1`, and `sample-subnet-is2`.

The following diagram describes the network topology:

```
IaaS network: sample-network
  |
  |_____ IaaS subnet: sample-subnet-shared
  |
  |_____ IaaS subnet: sample-subnet-is1
  |
  |_____ IaaS subnet: sample-subnet-is2
```

For more information about networks and subnets in GCP, see the Using Networks and Firewalls topic in the GCP documentation.

Step 2: Configure Networks for Routers

Navigate to the **Assign AZs and Networks** section of the PCF Isolation Segment tile to assign your isolation segment to the network you created in Step 1. See the Installing PCF Isolation Segment topic for more information.

Step 3: Configure Additional Routers

Navigate to the **Resource Config** section of the PCF Isolation Segment tile and use the dropdown menu to set your **Router** instances to a number greater than zero. See the Installing PCF Isolation Segment topic for more information.

Step 4: Add Routers to Load Balancer

If your IaaS supports it, navigate to the **Resource Config** section of the PCF Isolation Segment tile and enter the name of your load balancer under **Load Balancers**. See the documentation specific to your IaaS in Installing Pivotal Cloud Foundry for more information. If your IaaS does not support this configuration, you must create static IP addresses and assign them to your load balancer out of band.

Step 5: Configure DNS and Load Balancers

Create a separate domain name for each router instance group, and configure DNS to resolve these domain names to a load balancer that routes requests to the matching routers.

Note: You must configure your load balancers to forward requests for a given domain to one router instance group only.

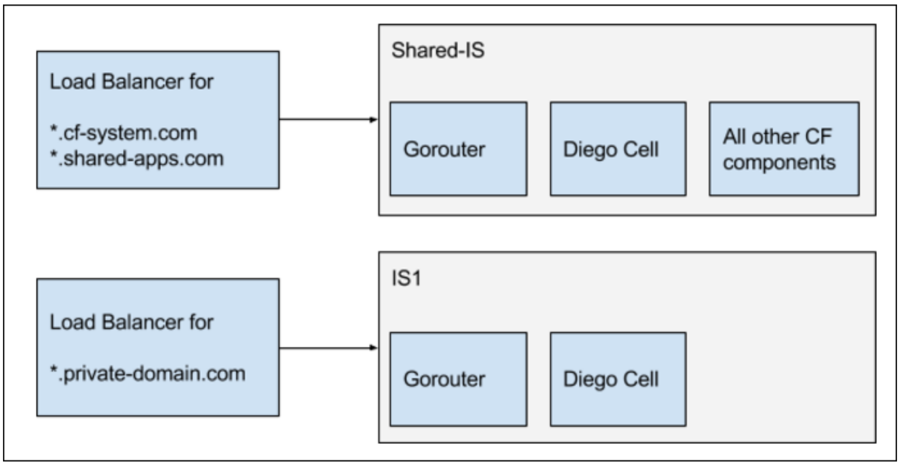
As router instance groups may be responsible for separate isolation segments, and an application may be deployed to only one isolation segment, requests should only reach a router that has access to the applications for that domain name. Load balancing requests for a domain across more than router instance group can result in request failures unless all the router instance groups have access to the isolation segments where applications for that domain are deployed.

The sections below describe a configuration that routes applications from distinct domain names, and a configuration that routes applications from a shared domain name. The diagrams illustrate a topology with separate load balancers, but you could also use one load balancer with multiple interfaces.

Distinct Domain Names

In a configuration that routes applications from distinct domain names:

- Requests for system domain `*.cf-system.com` and the shared domain `*.cf-apps.com` are forwarded to the routers for the shared isolation segment.
- Requests for private domain `*.private-domain.com` are forwarded to the routers for IS1.

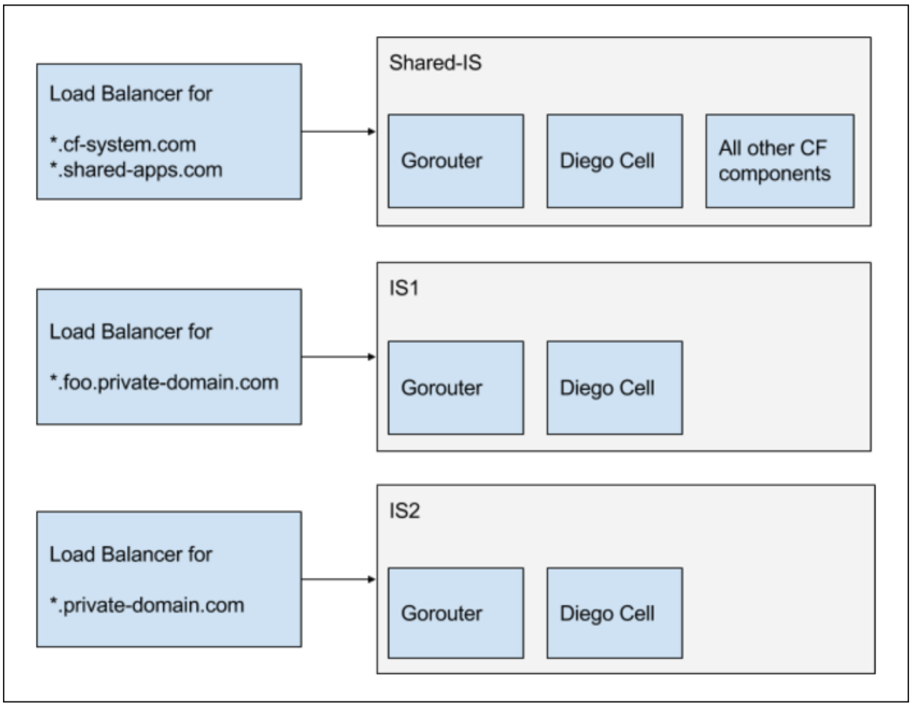


Shared Domain Name

It is a common requirement for applications on separate isolation segments to be accessible at domain names that share a domain, such as `private-apps.com`. To achieve this configuration while also obeying the guideline for forwarding requests for a domain to only one router instance group, create a new Cloud Foundry domain for a needed subdomain, such as `*.foo.private-apps.com.`

In this configuration:

- Requests for system domain `*.cf-system.com` and the shared domain `*.cf-apps.com` are forwarded to the routers for the shared isolation segment.
- Requests for private domain `*.foo.private-domain.com` are forwarded to the routers for IS1. Requests for private domain `*.private-domain.com` are forwarded to the routers for IS2.



Step 6: Configure Firewall Rules

Configure firewall rules to allow necessary traffic between the shared isolation segments (`sample-shared-is`) and the private isolation segments (`sample-is1` and `sample-is2`). Assuming a default deny-all rule, these rules should prevent a request with a spoofed Host header from being forwarded by a router to an application in another isolation segment.

Rule Name	Source subnetwork	Allowed Protocols/Ports	Target Tags	Reason

`shared-to-bosh`	`sample-subnet-shared`	`tcp:22,6868,25555,4222,25250`	`sample-bosh`	BOSH Agent on VMs in `sample-shared-is` to reach BOSH Director
`shared-internal`	`sample-subnet-shared`	`tcp`	`sample-shared-is`	VMs within `sample-shared-is` to reach one another
`shared-to-is1`	`sample-subnet-shared`	`tcp:1801`	`sample-is1`	Diego BBS in `sample-shared-is` to reach Cells in `sample-is1`
`shared-to-is2`	`sample-subnet-shared`	`tcp:1801`	`sample-is2`	Diego BBS in `sample-shared-is` to reach Cells in `sample-is2`
`is1-to-bosh`	`sample-subnet-is1`	`tcp:22,6868,25555,4222,25250`	`sample-bosh`	BOSH agent on VMs in `sample-is1` to reach BOSH Director
`is1-internal`	`sample-subnet-is1`	`tcp`	`sample-is1`	VMs within `sample-is1` to reach one another

`is1-to-shared`	`sample-subnet-is1`	`tcp:9090,9091,8082,8300,8301,8302,8889,8443,3000,4443,8080,3457,9023,9022,4222,8844`
		`udp:8301,8302,8600`
		
		See Port Reference Table for information about the processes that use these ports and their corresponding manifest properties.
			`sample-shared-is`	Diego Cells in `sample-is1` to reach BBS, Auctioneer, and CredHub in `sample-shared-is`, Consul Agent to reach Consul, Metron Agent to reach Traffic Controller, and Routers to reach NATS, UAA, and Routing API

`is2-to-bosh`	`sample-subnet-is2`	`tcp:22,6868,25555,4222,25250`	`sample-bosh`	BOSH agent on VMs in `sample-is2` to reach BOSH Director
`is2-internal`	`sample-subnet-is2`	`tcp`	`sample-is2`	VMs within `sample-is2` to reach one another

`is2-to-shared`	`sample-subnet-is2`	`tcp:9090,9091,8082,8300,8301,8302,8889,8443,3000,4443,8080,3457,9023,9022,4222,8844`
		`udp:8301,8302,8600`
		
		See Port Reference Table for information about the processes that use these ports and their corresponding manifest properties.
			`sample-shared-is`	Diego Cells in `sample-is2` to reach BBS, Auctioneer, and CredHub in `sample-shared-is`, Consul Agent to reach Consul, Metron Agent to reach Traffic Controller, and Routers to reach NATS, UAA, and Routing API

For more information consult the following topics:

- The `bosh-deployment` GitHub repository contains documentation describing ports used by agents to communicate with BOSH.
- The “Using Subnetworks” topic in the GCP documentation describes networks and firewall rules in the “Isolation of Subnetworks” section.

Port Reference Table

See the following table to understand which protocols and ports map to which processes and manifest properties for the `is#-to-shared` rules above.

Protocol	Port	Process	Manifest Property
`tcp`	`3000`	Routing API	`routing_api.port`
`tcp`	`3457`	Doppler	`metron_endpoint.dropsonde_port`
`tcp`	`4003`	VXLAN Policy Agent	`cf_networking.policy_server.internal_listen_port`
`tcp`	`4103`	Silk Controller	`cf_networking.silk_controller.listen_port`
`tcp`	`4222`	NATS	`router.nats.port`
`tcp`	`8080`	Diego file server	`diego.file_server.listen_addr`

`tcp`	`8082`	Doppler gRPC	`loggregator.doppler.grpc_port`
`tcp`	`8300`	Consul**	
`tcp`	`8301`	Consul**	
`tcp`	`8302`	Consul**	
`tcp`	`8443`	UAA	`uaa.ssl.port`
`tcp`	`8844`	CredHub	`credhub.port`
`tcp`	`8889`	Diego BBS	`diego.rep.bbs.api_location`
`tcp`	`8891`	Diego Database (Locket)	`diego.locket.listen_addr`
`tcp`	`9022`	CC stager	`capi.stager.cc.external_port`
`tcp`	`9023`	CC TPS	`capi.tps.cc.external_port`
`tcp`	`9090`	CC uploader	`http_port`
`tcp`	`9091`	CC uploader	`https_port`
`udp`	`8301`	Consul**	

** Consul documentation: Ports Used

Additional GCP Information

For more information, see the following:

- "Backend Services" in the GCP documentation
- BOSH Google Compute Engine CPI GitHub repository

Sharding Routers for Isolation Segments

You can configure router sharding for isolation segments depending on your use case:

Use Case	Description	How to Configure
Securing apps that run in an isolation segment	To provide security guarantees in addition to the firewall rules described above, you can configure sharding of the Gorouter's routing table, resulting in a router dedicated for an isolation segment having knowledge only of routes for applications in the same isolation segment.	1. In the **Networking** configuration pane of the Pivotal Application Service (PAS) tile, select the checkbox labeled **Routers reject requests for Isolation Segments**. 2. Set the **Router Sharding Mode** in the isolation segment tile to **Isolation Segment Only**.
Deploying additional routers for PAS	The flexibility of the configuration also supports deployment of a router that excludes all isolation segments.	1. In the **Networking** configuration pane of the PAS tile, select the checkbox labeled **Routers reject requests for Isolation Segments**. 2. Set the **Router Sharding Mode** in the isolation segment tile to **No isolation Segment**.

Note: For compute isolation only, you can leave the**Routers reject requests for isolation segments** checkbox unselected in the PAS **Networking** pane. This is the default setting, which does not require any additional routers for the Isolation Segment tile.

Metrics for Routers Associated with Isolation Segments

For metrics emitted by the Gorouter, metrics can be distinguished by the name of the job. For example, the following line is a metric emitted or `uptime` :

```
origin:"gorouter" eventType:ValueMetric timestamp:1491338040750977602 deployment:"superman.cf-app.com" job:"router_is1" index:"9a4b639c-8f0e-4b2b-b332-4161ee4646e6" ip:"10.0.16.23" valueMetric:<r
```

Using Feature Flags

Page last updated:

This topic describes how Cloud Foundry (CF) administrators can set feature flags using the Cloud Foundry Command Line Interface (cf CLI) to enable or disable the features available to users.

View and Edit Feature Flags

To perform the following procedures, you must be logged in to your deployment as an administrator using the cf CLI.

1. Use the `cf feature-flags` command to list the feature flags:

```
$ cf feature-flags

Features                              State
user_org_creation                     disabled
private_domain_creation               enabled
app_bits_upload                       enabled
app_scaling                           enabled
route_creation                        enabled
service_instance_creation             enabled
diego_docker                          disabled
set_roles_by_username                 enabled
unset_roles_by_username               enabled
task_creation                         enabled
env_var_visibility                    enabled
space_scoped_private_broker_creation  enabled
space_developer_env_var_visibility    enabled
service_instance_sharing              disabled
```

   For descriptions of the features enabled by each feature flag, see the Feature Flags section below.

2. To view the status of a feature flag, use the `cf feature-flag FEATURE-FLAG-NAME` command:

```
$ cf feature-flag user_org_creation

Retrieving status of user_org_creation as admin...
OK

Features           State
user_org_creation  disabled
```

3. To enable a feature flag, use the `cf enable-feature-flag FEATURE-FLAG-NAME` command:

```
$ cf enable-feature-flag user_org_creation
```

4. To disable a feature flag, use the `cf disable-feature-flag FEATURE-FLAG-NAME` command:

```
$ cf disable-feature-flag user_org_creation
```

Feature Flags

Only administrators can set feature flags. All flags are enabled by default except `user_org_creation` and `diego_docker`. When disabled, these features are only available to administrators.

The following list provides descriptions of the features enabled or disabled by each flag, and the minimum Cloud Controller API (CC API) version necessary to use the feature:

- `user_org_creation`: Any user can create an organization. If enabled, this flag activates the **Create a New Org** link in the dropdown menu of the left navigation menu in Apps Manager. Minimum CC API version: 2.12.
- `private_domain_creation`: An Org Manager can create private domains for that organization. Minimum CC API version: 2.12.
- `app_bits_upload`: Space Developers can upload app bits. Minimum CC API version: 2.12.

- `app_scaling` : Space Developers can perform scaling operations (i.e. change memory, disk, or instances). Minimum CC API version: 2.12.
- `route_creation` : Space Developers can create routes in a space. Minimum CC API version: 2.12.
- `service_instance_creation` : Space Developers can create service instances in a space. Minimum CC API version: 2.12.
- `diego_docker` : Space Developers can push Docker apps. Minimum CC API version 2.33.
- `set_roles_by_username` : Org Managers and Space Managers can add roles by username. Minimum CC API version: 2.37.
- `unset_roles_by_username` : Org Managers and Space Managers can remove roles by username. Minimum CC API version: 2.37.
- `task_creation` : Space Developers can create tasks on their application. Minimum CC API version: 2.47.
- `env_var_visibility` : All users can view environment variables. Minimum CC API version: 2.58.
- `space_scoped_private_broker_creation` : Space Developers can create space-scoped private service brokers. Minimum CC API version: 2.58.
- `space_developer_env_var_visibility` : Space Developers can view their v2 environment variables. Org Managers and Space Managers can view their v3 environment variables. Minimum CC API version: 2.58.
- `service_instance_sharing` : Space Developers can share service instances between two spaces (across orgs) in which they have the Space Developer role.

For more information about feature flag commands, see the **Feature Flags** section of the Cloud Foundry API documentation.

Stopping and Starting Virtual Machines

Page last updated:

This topic assumes you are using [BOSH CLI version 2](#).

This topic describes stopping and starting the component virtual machines (VMs) that make up a Pivotal Cloud Foundry (PCF) deployment.

In some cases you may want to stop all your VMs (for example, power down your deployment) or start all of your PAS VMs (for example, recover from a power outage.) You can stop or start all PAS VMs with a single `bosh` command.

If you want to shut down or start up a single VM in your deployment, you can use the manual process described on this page.

This procedure uses the BOSH Command Line Interface (BOSH CLI). See Advanced Troubleshooting with the BOSH CLI for more information about using this tool.

Stopping and Starting All PAS VMs

This section describes how to stop and start all the VMs in your deployment.

Stopping All PAS VMs

To shut down all the VMs in your deployment, perform the following steps:

1. Scale down the following jobs to one instance:
   - consul_server
   - mysql

2. Run the following command for each of the deployments listed in the previous step:

   ```
   bosh -e MY-ENV -d MY-DEPLOYMENT stop --hard
   ```

   Replace the text above with the following:
   - `MY-ENV`: the alias you set for the BOSH Director.
   - `MY-DEPLOYMENT`: the name of your deployment.

   For example:

   ```
   $ bosh -e prod -d mysql stop --hard
   ```

   This command stops all VMs in the specified deployment. The `--hard` flag instructs BOSH to delete the VMs but retain any persistent disks.

Starting All PAS VMs

Perform the following steps to start all the VMs in your deployment:

1. Select the product deployment for the VMs you want to shut down. You can run the following command to locate CF deployment manifests:

   ```
   $ find /var/tempest/workspaces/default/deployments -name cf-*.yml
   ```

2. Run the following command:

   ```
   bosh -e MY-ENV -d MY-DEPLOYMENT start
   ```

   Replace the text above with the following:
   - `MY-ENV`: the alias you set for the BOSH Director.
   - `MY-DEPLOYMENT`: the name of your deployment.

   For example:

```
$ bosh -e prod -d mysql start
```

This command starts all VMs in the specified deployment.

3. If you require high availability in your deployment, scale up all instance groups to the original or desired counts.

Stopping and Starting Individual PAS VMs

This section describes how to stop and start individual VMs in your deployment.

Find the Names of Your PAS Virtual Machines

You need the full names of the VMs to stop and start them using the BOSH CLI. To find full names for the VMs running each component, run `bosh -e MY-ENV instances`, replacing `MY-ENV` with the alias you set for your BOSH Director. To filter the list of instances by deployment, run `bosh -e MY-ENV -d MY-DEPLOYMENT instances`.

For example:

```
$ bosh -e prod -d mysql instances
...
Deployment 'mysql'

Instance                    Process State  AZ  IPs
mysql/0123-abcd-4567ef89    running        -   10.244.0.6
mysql/abcd-0123-ef4567ab    running        -   10.244.0.2

2 instances
...
```

You can see the full name of each VM in the `Instance` column of the terminal output. Each full name includes:

- A prefix indicating the component function of the VM.
- An identifier string specific to the VM.

For any component, you can look for its prefix in the `bosh instances` output to find the full name of the VM or VMs that run it.

Stopping an Individual PAS VM

To stop a job, run the following command for the component in your PAS deployment, replacing `MY-ENV` with the alias you set for your BOSH Director and `MY-DEPLOYMENT` with the name of the deployment:

```
bosh -e MY-ENV -d MY-DEPLOYMENT stop VM-NAME
```

To delete the instance that contains the job, run the following command for the component in your PAS deployment:

```
bosh -e MY-ENV -d MY-DEPLOYMENT stop VM-NAME --hard
```

Use the full name of the component VM as listed in your `bosh instances` terminal output without the unique identifier string.

For example, the following command stops the Loggregator Traffic Controller job:

```
$ bosh -e prod -d loggregator stop loggregator_trafficcontroller
```

To stop a specific instance of a job, include the identifier string at the end of its full name.

For example, the following command stops the Loggregator Traffic Controller job on only one Diego cell instance:

```
$ bosh -e prod -d loggregator stop loggregator_trafficcontroller/0123-abcd-4567ef89
```

To delete the VM, include `--hard` at the end of the command. This command does not delete persistent disks.

For example, the following command deletes a specific Loggregator Traffic Controller instance:

```
$ bosh -e prod -d loggregator stop loggregator_trafficcontroller/0123-abcd-4567ef89 --hard
```

Starting an Individual PAS VM

Run the following command for the component in your PAS deployment you wish to start, replacing `MY-ENV` with the alias you set for your BOSH Director and `MY-DEPLOYMENT` with the name of the deployment. Use the full name of the component VM as listed in your `bosh vms` terminal output without the unique identifier string.

```
bosh -e MY-ENV -d MY-DEPLOYMENT start VM-NAME
```

The following example command starts the Loggregator Traffic Controller VM:

```
$ bosh -e prod -d loggregator start loggregator_trafficcontroller
```

To start a specific instance of a VM, include the identifier string at the end of its full name.

For example, the following command starts the Loggregator Traffic Controller job on one Diego cell instance:

```
$ bosh -e prod -d loggregator start loggregator_trafficcontroller/0123-abcd-4567ef89
```

Managing Diego Cell Limits During Upgrade

Cloud Foundry supports rolling upgrades in high availability environments. A rolling upgrade means that you can continue to operate an existing Cloud Foundry deployment and its running app instances while upgrading the platform.

Note: Rolling upgrade is not available in your deployment if you have not configured your deployment to be highly available. See High Availability in Cloud Foundry.

Diego Cell Upgrade Process

To upgrade Cloud Foundry, BOSH must drain all Diego cell VMs that host app instances. BOSH manages this process by upgrading a batch of cells at a time.

The number of cells that undergo upgrade simultaneously (either in a state of shutting down or coming back online) is controlled by the `max_in_flight` value of the Diego cell job. For example, if `max_in_flight` is set to `10%` and your deployment has 20 Diego cell job instances, then the maximum number of cells that BOSH can upgrade at a single time is `2`.

When BOSH triggers an upgrade, each Diego cell undergoing upgrade enters "evacuation" mode. Evacuation mode means that the cell stops accepting new work and signals the rest of the Diego system to schedule replacements for its app instances. This scheduling is managed by the Diego auctioneer process.

The evacuating cells continue to interact with the Diego system as replacements come online. The cell undergoing upgrade waits until either its app instance replacements run successfully before shutting down the original local instances, or for the evacuation process to time out. This "evacuation timeout" defaults to 10 minutes.

If cell evacuation exceeds this timeout, then the cell stops its app instances and shuts down. The Diego system continues to re-emit start requests for the app replacements.

Preventing Overload

A potential issue arises if too many app instance replacements are slow to start or do not start successfully at all.

If too many app instances are started at once, then the load of these starts on the rest of the system can cause other applications that are already running to crash and be rescheduled. These events can result in a cascading failure.

To prevent overload, Cloud Foundry provides two major throttle configurations:

- **The maximum number of starting containers that Diego can start in Cloud Foundry.** This is a deployment-wide limit. The default value and ability to override this configuration depends on the version of Cloud Foundry deployed. For information about how to configure this setting, see the Setting a Maximum Number of Started Containers topic.
- **The `max_in_flight` setting for the Diego cell job configured in the BOSH manifest.** This configuration, expressed as a percentage or an integer, sets the maximum number of job instances that can be upgraded simultaneously. For example, if your deployment is running 10 Diego cell job instances and the configured `max_in_flight` value is `20%`, then only 2 Diego cell job instances can start up at a single time.

  To retrieve or override the existing `max_in_flight` value in Ops Manager Director, use the Ops Manager API. See the Ops Manager API documentation provided with your Ops Manager installation at `https://YOUR-OPSMAN-FQDN/docs/`.

The values of the above throttle configurations depend on the version of PCF that you have deployed and whether you have overridden the default values.

Refer to the following table for existing defaults and, if necessary, determine the override values in your deployment.

PCF Version	Starting Container Count Maximum	Starting Container Count Overridable?	Maximum In Flight Diego Cell Instances	Maximum In Flight Diego Cell Instances Overridable?
PCF 1.7.43 and earlier	No limit set	No	1 instance	No
PCF 1.7.44 to 1.7.49	200	No	1 instance	No
PCF 1.7.50 +	200	No	1 instance	No
PCF 1.8.0 to 1.8.29	No limit set	No	10% of total instances	No

PCF 1.8.30 +	200	Yes	10% of total instances	No
PCF 1.9.0 to 1.9.7	No limit set	No	4% of total instances	Yes
PCF 1.9.8 +	200	Yes	4% of total instances	Yes
PCF 1.10.0 and later	200	Yes	4% of total instances	Yes
PCF 1.12.0 and later	200	Yes	4% of total instances	Yes

Setting a Maximum Number of Started Containers

Page last updated:

This topic describes how to use the auctioneer job to configure the maximum number of app instances running at a given time. This prevents Diego from scheduling too much work for your platform to handle. A lower default can prevent server overload, which may be important if you're using a small server.

The auctioneer only schedules a fixed number of app instances to start at one time. This limit applies to both single and multiple Diego Cells. For example, if you set the limit to five started instances, it does not matter if you have one Diego Cell with ten instances or five Diego Cells with two instances each. The auctioneer will not allow more than five instances to start at the same time.

If you are using a cloud-based IaaS, rather than a smaller on-premise solution, Pivotal recommends setting a larger default. By default, the maximum number of started instances is 200.

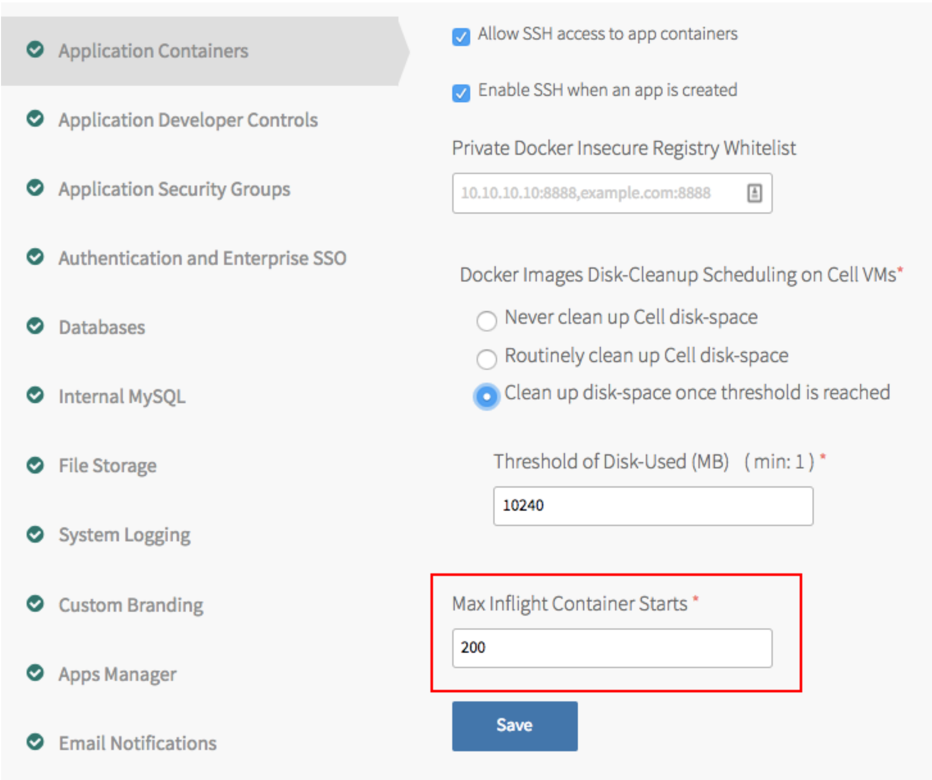
You can configure the maximum number of started instances in the Settings tab of the Pivotal Application Service (PAS) tile.

1. Log in to Operations Manager.

2. Click the PAS tile.

3. Click **Application instances** in the sidebar.

4. In the Max Inflight Container Starts field, type the maximum number of started instances.



5. Click **Save**.

Enabling IPv6 for Hosted Applications

Page last updated:

The procedure described below allows apps deployed to Pivotal Application Service to be reached using IPv6 addresses.

> **Note**: Amazon Web Services (AWS) EC2 instances currently do not support IPv6.

Pivotal Application Service system components use a separate DNS subdomain from hosted applications. These components currently support only IPv4 DNS resolved addresses. This means that although an IPv6 address can be used for application domains, the system domain must resolve to an IPv4 address.

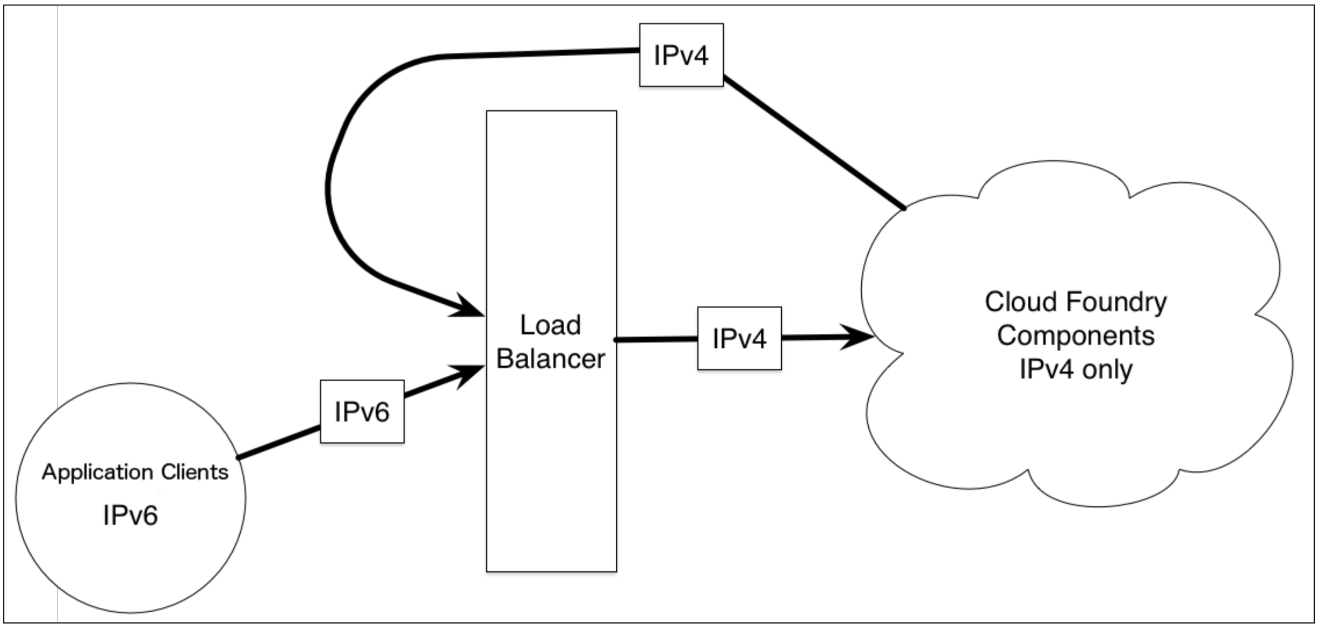
Complete the following steps to enable support for IPv6 application domains:

1. Set up an external load balancer for your Pivotal Application Service deployment. See Using Your Own Load Balancer.
2. Configure DNS to resolve application domains to an IPv6 address on your external load balancer.

   > **Note**: Your IPv4 interface for the system domain and IPv6 interface for application domain can be configured on the same or different load balancers.

3. Configure the external load balancer to route requests for an IPv6 address to an IPv4 address as follows:
   - If you are using the HAProxy load balancer for SSL termination, route to its IPv4 address.
   - Otherwise, route directly to the IPv4 addresses of the Gorouters.

The following diagram illustrates how a single load balancer can support traffic on both IPv4 and IPv6 addresses for a Pivotal Application Service installation.



See Routes and Domains for more information about domains in Pivotal Application Service.

Securing Traffic into Cloud Foundry

Page last updated:

This topic describes the options for securing HTTP traffic into your Pivotal Application Service deployment with TLS certificates. You can configure the location where your deployment terminates TLS depending on your needs and certificate restrictions.

Protocol Support

The Gorouter supports HTTP/HTTPS requests only. For more information about features supported by the Gorouter, see the HTTP Routing topic.

To secure non-HTTP traffic over TCP Routing, terminate TLS at your load balancer or at the application. See TCP Routing for details.

TLS Termination Options for HTTP Routing

There are several options for terminating TLS for HTTP traffic. You can terminate TLS at the Gorouter, your load balancer, or at both.

The following table summarizes TLS termination options and which option to choose for your deployment.

If the following applies to you:	Then configure TLS termination at:	Related topic and configuration procedure
• You want the optimum balance of performance and security, and • You want to make minimum changes to your load balancer, or • You are deploying CF to AWS. For information about AWS limitations, see TLS Cipher Suite Support by AWS ELBs.	Gorouter only	Terminating TLS at the Gorouter Only
• You require TLS termination at a load balancer, or • You want the highest level of security, and • You do not mind a slightly less performant deployment.	Load Balancer and Gorouter	Terminating TLS at the Load Balancer and Gorouter
• You require TLS termination at a load balancer, and • You prefer unencrypted traffic between the Load Balancer and the Gorouter.	Load Balancer only	Terminating TLS at the Load Balancer Only
Optionally, if you are deploying HAProxy, and	**Then in addition, terminate SSL/TLS at:**	**Related topic and configuration procedure**
• You would like to secure traffic to the HAProxy.	HAProxy	Terminating SSL/TLS at HAProxy

Certificate Requirements

The following requirements apply to the certificates you use to secure traffic into PAS

- You must obtain at least one TLS certificate for your environment.
  - In a production environment, use a signed TLS certificate (trusted) from a known certificate authority (CA).
  - In a development or testing environment, you can use a trusted CA certificate or a self-signed certificate. You can generate a self-signed certificate with `openssl` or a similar tool.

  > Alternately, you can use the PAS Ops Manager interface to generate a certificate for you. Certificates generated in PAS are signed by the Ops Manager Certificate Authority. They are not technically self-signed, but they are sometimes referred to as "Self-Signed Certificates" in the Ops Manager UI and throughout this documentation.

- Certificates used in CF must be encoded in the PEM format.
- The Gorouter supports mutual TLS, and validates a client provided certificate chain against its CA certificates if one is provided in the TLS handshake, but does not require it. Depending on whether you choose to terminate at both the Load Balancer and the Gorouter, or at the Gorouter alone, the client certificate may be that of the load balancer or of the originating client.
- The certificate on the Gorouter must be associated with the correct hostname so that HTTPS can validate the request.

- If wildcard certificates are not supported for some or all of your domains, then configure termination requests at the load balancer only. In this type of deployment, the load balancer passes unencrypted traffic to the Gorouter. As a result, you avoid having to reissue and reinstall certificates on the Gorouter for every app or UAA security zone.
- Extended Validation (EV) certificates support multiple hostnames, like SAN, but do not support wildcards. As the Gorouter has not been tested with EV certificates, if EV certificates are required, then terminate TLS at the load balancer only.
- Given the dynamic and multi-tenant nature of PAS, use of wildcard domains is highly recommended to avoid the need for adding an additional certificate for each application.

Multiple Certificates

In order to support custom domains on CF, an operator has to configure the Gorouter with a certificate that represents the domain. It is recommended that operators add a new certificate instead of reissuing a single certificate when adding TLS support for an additional domain. Using multiple certificates provides a security benefit in that it prevents clients from discovering all the custom domains of applications running on a CF platform.

The Gorouter supports SNI and can be configured with multiple certificates, each which may optionally include wildcard and alternative names. The Gorouter uses SNI to determine the correct certificate to present in a TLS handshake. It requires clients to support the SNI protocol by sending a server name outside the encrypted request payload. For clients that do not support SNI, the Gorouter presents a default certificate. The default is the first certificate keypair in the Gorouter's configuration.

The Gorouter decides which certificate to provide in the TLS handshake as follows:

- If a client provides an SNI header with a ServerName that matches to a configured certificate keypair, the Gorouter returns the matching certificate.
- If a client provides an SNI header with a ServerName that does not match a configured certificate keypair, the Gorouter returns the default certificate.

The first certificate keypair listed is used as the default.

The Gorouter supports both RSA and ECDSA certificates in PEM encoding. In the case that a certificate chain is required, the order should be as follows: primary certificate, intermediate certificate, then root certificate.

How to Configure Multiple Certificate Keypairs

To configure multiple HTTPS certificate keypairs for PAS, add each keypair along with a meaningful name in the applicable **Certificates and Private Keys for HAProxy and Router** fields of the **Networking** configuration screen in PAS. For more information, see the Deploying PAS topic for your platform. For example, see Deploying PAS on GCP if you are using GCP.

In PCF, multiple certificates configured for the Gorouter are also configured for HAProxy.

TLS Cipher Suite Support

Some CF components like the Gorouter support additional TLS cipher suites to accommodate older clients. As a security best practice, only configure the TLS cipher suites that you need for your deployment.

Default Gorouter Cipher Suites

By default, the Gorouter supports the following TLS cipher suites, both of which require TLS v1.2:

RFC	OpenSSL
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	ECDHE-RSA-AES128-GCM-SHA256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDHE-RSA-AES256-GCM-SHA384

You can override the default cipher suites in the **TLS Cipher Suites for Router** field in the **Networking** tab of the PAS tile. See the following procedures for either Gorouter Only or Load Balancer and Gorouter for more information about using custom SSL ciphers.

TLS Cipher Suite Support by AWS Load Balancers

AWS Classic Load Balancers (formerly referred to as ELBs) support configuration of cipher suites for front-end connections with clients only. When configuring Classic Load Balancers to forward requests to Gorouters over TLS, operators may encounter a "Cipher Suite Mismatch" error. This is because the cipher suites supported by Classic Load Balancers for TLS handshakes with backends (Gorouters in this case) are hardcoded, undocumented, and do not support the Gorouter default cipher suites.

Operators have three options:

- Configure Classic Load Balancer listeners in TCP mode so that TCP connections from clients are passed through the Classic Load Balancer to Gorouters on port 443. Then Gorouters are the first point of TLS termination. This option is the recommended solution.
- If you require TLS termination at an AWS load balancer in addition to terminating at the Gorouter, use AWS Application Load Balancers (ALBs) that support the Gorouter default cipher suites.
- Configure the Gorouter to support an older TLS version (v1.1 or v1.0) and then configure any of the TLS v1.0 or v1.1 cipher suites listed below in addition to the defaults. This option is the least secure option and should be used only if you are required to use an AWS Classic Load Balancer and also require TLS termination at both the load balancer and the Gorouter.

  To add an older cipher suite to the Gorouter cipher suites, perform the following steps:
  1. In the **Networking** configuration screen of PAS, select either TLS v1.0 or v1.1 in the **Minimum version of TLS supported by HAProxy and Router field**.
  2. Add one or more of the Gorouter-supported cipher suites listed in TLS v1.0 and v1.1 to the existing list of default cipher suites in the **TLS Cipher Suites for Router** field.

  For more information, see the PAS configuration documentation for your IaaS. For example, if you are deploying PCF on GCP, then see Deploying PAS on GCP.

  Since this option relies on using an older cipher suite, consult your security team before implementing this solution.

TLS v1.2

The following cipher suites are optionally supported for TLS v1.2 only:

RFC	OpenSSL
TLS_RSA_WITH_AES_128_GCM_SHA256	AES128-GCM-SHA256
TLS_RSA_WITH_AES_256_GCM_SHA384	AES256-GCM-SHA384
TLS_ECDHE_ECDSA_WITH_RC4_128_SHA	ECDHE-ECDSA-RC4-SHA
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	ECDHE-ECDSA-AES128-SHA
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	ECDHE-ECDSA-AES256-SHA
TLS_ECDHE_RSA_WITH_RC4_128_SHA	ECDHE-RSA-RC4-SHA
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	ECDHE-RSA-DES-CBC3-SHA
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	ECDHE-RSA-AES128-SHA
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	ECDHE-RSA-AES256-SHA
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	ECDHE-RSA-AES128-GCM-SHA256
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	ECDHE-ECDSA-AES128-GCM-SHA256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDHE-RSA-AES256-GCM-SHA384
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	ECDHE-ECDSA-AES256-GCM-SHA384
TLS_RSA_WITH_AES_128_CBC_SHA256	AES128-SHA256
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	ECDHE-ECDSA-AES128-SHA256
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	ECDHE-RSA-AES128-SHA256
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305	ECDHE-RSA-CHACHA20-POLY1305
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305	ECDHE-ECDSA-CHACHA20-POLY1305

TLS v1.0 and v1.1

The following cipher suites are optionally supported for TLS v1.0 and TLS v1.1 only:

RFC	OpenSSL
TLS_RSA_WITH_RC4_128_SHA	RC4-SHA

TLS_RSA_WITH_3DES_EDE_CBC_SHA	DES-CBC3-SHA
TLS_RSA_WITH_AES_128_CBC_SHA	AES128-SHA
TLS_RSA_WITH_AES_256_CBC_SHA	AES256-SHA

You can override the default cipher suites in the **TLS Cipher Suites for Router** field in the **Networking** tab of the PAS tile. See the following procedures for either Gorouter Only or Load Balancer and Gorouter for more information about using custom SSL ciphers.

See Golang Constants and OpenSSL Cipher Suites for more information about supported ciphers.

Note: ECDSA ciphers require a certificate and key for DSA, as opposed to RSA.

Mutual Authentication with Clients

Gorouter supports validation of client certificates in TLS handshakes with clients, also known as mutual authentication. Operators can choose whether Gorouter requests client certificates and when requesting certificates, whether or not to require them.

By default, Gorouter requests but does not require client certificates in TLS handshakes.

To configure Gorouter behavior for handling client certificates, select one of the options in the **Router behavior for Client Certificates** field of the **Networking** configuration screen in PAS.

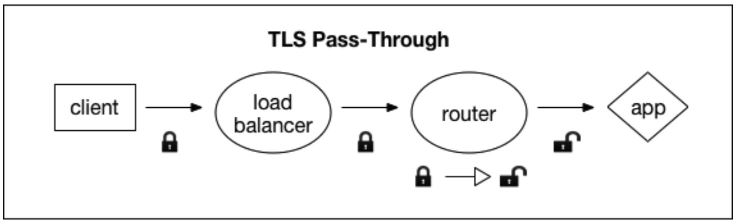
- **Router does not request client certificates.** This option is incompatible with the XFCC configuration options **TLS terminated for the first time at HAProxy** and **TLS terminated for the first time at the Router** in PAS because these options require mutual authentication. As client certificates are not requested, client will not provide them, and thus validation of client certificates will not occur.
- **Router requests but does not require client certificates.** Gorouter requests client certificates in TLS handshakes, validates them when presented, but does not require them. This is the default configuration.
- **Router requires client certificates.** Gorouter validates that the client certificate is signed by a Certificate Authority that Gorouter trusts. If Gorouter cannot validate the client certificate, the TLS handshake fails.

Terminating TLS at the Gorouter Only

In this configuration, the load balancer does not terminate TLS for CF domains at all. Instead, it passes through the underlying TCP connection to the Gorouter.

This option is the recommended and more performant option, establishing and terminating a single TLS connection.

The following diagram illustrates communication between the client, load balancer, Gorouter, and app.



Traffic between the load balancer and the Gorouter is encrypted only if the client request is encrypted.

About HTTP Header Forwarding

If you terminate TLS at the Gorouter only, your load balancer does not send HTTP headers.

The Gorouter appends the `X-Forwarded-For` and `X-Forwarded-Proto` headers to requests forwarded to applications and platform system components. `X-Forwarded-For` is set to the IP address of the source. Depending on the behavior of your load balancer, this may be the IP address of your load balancer. For the Gorouter to deliver the IP address of the client to applications, configure your load balancer to forward the IP address of the client or configure your load balancer to send the client IP address using the PROXY protocol. The Gorouter will set `X-Forwarded-Proto` to the scheme of the client request.

For more information on HTTP headers in CF, see HTTP Headers. If you are configuring the forwarding of client certificates, see Forward Client Certificate to Applications.

Procedure: Gorouter Only

Perform the following steps to configure SSL termination on the Gorouter in Pivotal Cloud Foundry (PCF):

1. Configure your load balancer to pass through TCP requests from the client to the Gorouter.
2. Navigate to the Ops Manager Installation Dashboard.
3. Click the Pivotal Application Service (PAS) tile in the Installation Dashboard.
4. Click **Networking**.
5. For PCF deployments on OpenStack or vSphere, choose IP addresses for the Gorouters from the subnet configured for Ops Manager and enter them in the **Router IPs** field. Then configure your load balancer to forward requests for the above domains to these IP addresses. For more information, see the PAS networking configuration topic for OpenStack or vSphere.
6. In the **Certificates and Private Keys for HAProxy and Router** field, click the **Add** button to define at least one certificate keypair for HAProxy and Router. For each certificate keypair that you add, assign a name, enter the PEM-encoded certificate chain and PEM-encoded private key. You can either upload your own certificate or generate an RSA certificate in PAS. For options and instructions on creating a certificate for your wildcard domains, see Creating a Wildcard Certificate for PCF Deployments.
7. In the **Minimum version of TLS supported by HAProxy and Router**, select the minimum version of TLS to use in Gorouter communications. The Gorouter uses TLS v1.2 by default. If you need to accommodate clients that use an older version of TLS, select a lower minimum version. For a list of TLS ciphers supported by the Gorouter, see Cipher Suites.
8. Under **HAProxy forwards requests to Router over TLS**, select **Disable**.
9. If you want to use a specific set of TLS ciphers for the Gorouter, configure **TLS Cipher Suites for Router**. Enter an ordered, colon-separated list of TLS cipher suites in the OpenSSL format. For example, if you have selected support for an earlier version of TLS, you can enter cipher suites supported by this version. For a list of TLS ciphers supported by the Gorouter, see Cipher Suites. Otherwise, leave the default values in this field.
10. (Optional) If you are not using SSL encryption or if you are using self-signed certificates, you can select **Disable SSL certificate verification for this environment**. Selecting this checkbox also disables SSL verification for route services.

    Use this checkbox only for development and testing environments. Do not select it for production environments.

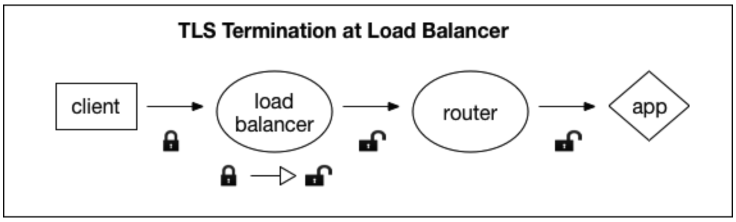
11. (Optional) If you do not want the Gorouter to accept any non-encrypted HTTP traffic, select the **Disable HTTP on HAProxy and Router** checkbox.
12. In the **Configure the CF Router support for the X-Forwarded-Client-Cert header** field, select the third option, **Strip the XFCC header when present and set it to the client certificate**.
13. Click **Save**.
14. In the PAS tile, click **Resource Config**.
15. In the **Instances** drop down for the **HAProxy** job, select `0` instances.
16. Click **Save**.

Terminating TLS at the Load Balancer Only

In this configuration, your load balancer terminates TLS, and passes unencrypted traffic to the Gorouter, which routes it to your app. Traffic between the load balancer and the Gorouter is not encrypted.

This option is recommended if you cannot use SAN certificates and if you do not require traffic to be encrypted between the load balancer and the Gorouter.

The following diagram illustrates communication between the client, load balancer, Gorouter, and app.



About HTTP Header Forwarding

If you terminate TLS at your load balancer, then you must also configure the load balancer to append the `X-Forwarded-For` and `X-Forwarded-Proto` HTTP headers to the HTTP traffic it passes to the Gorouter.

For more information on HTTP headers in CF, see HTTP Headers. If you are configuring the forwarding of client certificates, see Forward Client Certificate to Applications.

Procedure: Load Balancer Only

Perform the following steps to configure SSL termination on the load balancer only in Pivotal Cloud Foundry (PCF):

1. Create an A record in your DNS that points to your load balancer IP address. The A record associates the **System Domain** and **Apps Domain** that you configure in the **Domains** section of the Pivotal Application Service (PAS) tile with the IP address of your load balancer.

   For example, with `cf.example.com` as the main subdomain for your Cloud Foundry deployment and a load balancer IP address `198.51.100.1`, you must create an A record in your DNS that serves `example.com` and points `*.cf` to `198.51.100.1`.

Name	Type	Data	Domain
*.cf	A	198.51.100.1	example.com

2. Navigate to the Ops Manager Installation Dashboard.

3. Click the PAS tile in the Installation Dashboard.

4. Click **Networking**.

5. For PCF deployments on OpenStack or vSphere, choose IP addresses for the Gorouters from the subnet configured for Ops Manager and enter them in the **Router IPs** field. Then configure your load balancer to forward requests for the above domains to these IP addresses. For more information, see the PAS networking configuration topic for OpenStack or vSphere.

6. In the **Certificates and Private Keys for HAProxy and Router** field, click the **Add** button to define one certificate keypair for HAProxy and Router. Since you have opted for unencrypted traffic behind the load balancer, then you can generate an RSA certificate in PAS.

7. In the **Minimum version of TLS supported by HAProxy and Router**, select the minimum version of TLS to use in HAProxy communications. HAProxy use TLS v1.2 by default. If you need to accommodate clients that use an older version of TLS, select a lower minimum version. For a list of TLS ciphers supported by the HAProxy, see Cipher Suites.

8. Under **HAProxy forwards requests to Router over TLS**, select **Disable**.

9. If you want to use a specific set of TLS ciphers for HAProxy, configure **TLS Cipher Suites for HAProxy**. Enter an ordered, colon-separated list of TLS cipher suites in the OpenSSL format. For example, if you have selected support for an earlier version of TLS, you can enter cipher suites supported by this version. Otherwise, leave the default values in this field.

10. (Optional) If you are not using SSL encryption or if you are using self-signed certificates, you can select **Disable SSL certificate verification for this environment**. Selecting this checkbox also disables SSL verification for route services.

    Use this checkbox only for development and testing environments. Do not select it for production environments.

11. (Optional) If you do not want HAProxy or the Gorouter to accept any non-encrypted HTTP traffic, select the **Disable HTTP on HAProxy and Router** checkbox.

12. In the **Configure the CF Router support for the X-Forwarded-Client-Cert header** field, select **Always forward the XFCC header in the request, regardless of the whether the client connection is mTLS**.

13. Click **Save**.

14. After you complete the configuration in PCF, add your certificate or certificates to your load balancer, and configure its listening port. The procedures vary depending on your IaaS.

15. Configure your load balancer to append the `X-Forwarded-For` and `X-Forwarded-Proto` headers to client requests.

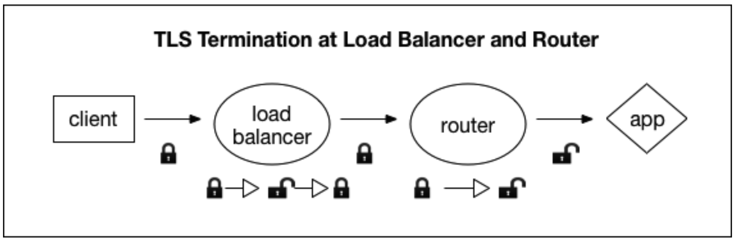
    If the load balancer cannot be configured to provide the `X-Forwarded-For` header, the Gorouter will append it in requests forwarded to applications and system components, set to the IP address of the load balancer.

    > **Note**: If the load balancer accepts unencrypted requests, it **must** provide the X-Forwarded-Proto header. Conversely, if the load balancer cannot be configured to send the X-Forwarded-Proto header, it should not accept unencrypted requests. Otherwise, applications and platform system components that require encrypted client requests will accept unencrypted requests when they should not accept them.

Terminating TLS at the Load Balancer and Gorouter

In this configuration two TLS connections are established: one from the client to the load balancer, and another from the load balancer to the Gorouter. This configuration secures all traffic between the load balancer and the Gorouter.

The following diagram illustrates communication between the client, load balancer, Gorouter, and app.



This option is less performant, but allows for termination at a load balancer, as well as secure traffic between the load balancer and the Gorouter.

Certificate Guidelines

In this deployment scenario, the following guidelines apply:

- Certificates for the PAS domains must be stored on the load balancer, as well as on the Gorouter.
- Generate certificates for your load balancer and the Gorouter with different keys. If the key for the certificate on the Gorouter is compromised, then the certificate on the load balancer is not at risk, and vice versa.
- If you choose to host only one certificate on the Gorouter and many on your load balancer, configure your load balancer with the CA and hostname with which to validate the certificate hosted by the Gorouter.

About Hostname Verification

Hostname verification between the load balancer and the Gorouter is unnecessary when the load balancer is already configured with the Gorouter's IP address to correctly route the request.

If the load balancer uses DNS resolution to route requests to the Gorouters, then you should enable hostname verification.

About HTTP Header Forwarding

If you terminate TLS at your load balancer, then you must configure the load balancer to append the `X-Forwarded-For` and `X-Forwarded-Proto` HTTP headers to requests it sends to the Gorouter.

If you terminate TLS at your load balancer but it does not support HTTP, such that it cannot append HTTP headers, a workaround exists. We recommend you use this workaround **only if your load balancer does not accept unencrypted requests**. Configure your load balancer to send the client IP address using the PROXY protocol, and enable PROXY in the Gorouter. As the `X-Forwarded-Proto` header will not be present, configure the Gorouter to force-set this header to 'HTTPS'.

For more information on HTTP headers in CF, see HTTP Headers. If you are configuring the forwarding of client certificates, see Forward Client Certificate to Applications.

Procedure: Load Balancer and Gorouter

Perform the following steps to configure SSL termination on the Gorouter and load balancer in Pivotal Cloud Foundry (PCF):

1. Create an A record in your DNS that points to your load balancer IP address. The A record associates the **System Domain** and **Apps Domain** that you configure in the **Domains** section of the Pivotal Application Service (PAS) tile with the IP address of your load balancer.

   For example, with `cf.example.com` as the main subdomain for your Cloud Foundry (CF) deployment and a load balancer IP address `198.51.100.1`, you must create an A record in your DNS that serves `example.com` and points `*.cf` to `198.51.100.1`.

Name	Type	Data	Domain
*.cf	A	198.51.100.1	example.com

2. Navigate to the Ops Manager Installation Dashboard.

3. Click the PAS tile in the Installation Dashboard.

4. Click **Networking**.

5. For PCF deployments on OpenStack or vSphere, choose IP addresses for the Gorouters from the subnet configured for Ops Manager and enter them in the **Router IPs** field. Then configure your load balancer to forward requests for the above domains to these IP addresses. For more information, see the PAS networking configuration topic for OpenStack or vSphere.

6. In the **Certificates and Private Keys for HAProxy and Router** field, click the **Add** button to define at least one certificate keypair for HAProxy and Router. For each certificate keypair that you add, assign a name, enter the PEM-encoded certificate chain and PEM-encoded private key. You can either upload your own certificate or generate an RSA certificate in PAS. For options and instructions on creating a certificate for your wildcard domains, see Creating a Wildcard Certificate for PCF Deployments.

7. In the **Minimum version of TLS supported by HAProxy and Router**, select the minimum version of TLS to use in HAProxy and Gorouter communications. The Gorouter use TLS v1.2 by default. If you need to accommodate clients that use an older version of TLS, select a lower minimum version. For a list of TLS ciphers supported by the Gorouter, see Cipher Suites.

8. If you are using **HAProxy**, complete the following steps:

   a. Under **HAProxy forwards requests to Router over TLS**, select **Enable**.
   b. In the **Certificate Authority for HAProxy Backend** field, specify the Certificate Authority (CA) that signed the certificate you configured in the **Certificate and Private Key for HAProxy and Router** field.

   > If you used the **Generate RSA Certificate** link to generate a self-signed certificate, then the CA to specify is the Ops Manager CA, which you can locate at the `/api/v0/certificate_authorities` endpoint in the Ops Manager API.

   c. If you want to use a specific set of TLS ciphers for HAProxy, configure **TLS Cipher Suites for HAProxy**. Enter an ordered, colon-separated list of TLS cipher suites in the OpenSSL format. For example, if you have selected support for an earlier version of TLS, you can enter cipher suites supported by this version. Otherwise, leave the default values in this field.
   d. In the **Configure the CF Router support for the X-Forwarded-Client-Cert header** field, select **Always forward the XFCC header in the request, regardless of the whether the client connection is mTLS**.
   e. Proceed to step 11.

9. If you want to use a specific set of TLS ciphers for the Gorouter, configure **TLS Cipher Suites for Router**. Enter an ordered, colon-separated list of TLS cipher suites in the OpenSSL format. For example, if you have selected support for an earlier version of TLS, you can enter cipher suites supported by this version. For a list of TLS ciphers supported by the Gorouter, see Cipher Suites. Otherwise, leave the default values in this field.

10. If you are not using HAProxy, complete the following steps:

    a. Under **HAProxy forwards requests to Router over TLS**, select **Disable**.
    b. In the **Configure the CF Router support for the X-Forwarded-Client-Cert header** field, select any of the available options depending on your client application needs. For more information about XFCC header forwarding, see Forward Client Certificate to Applications.
    c. In the PAS tile, click **Resource Config**.
    d. In the **Instances** drop down for the **HAPRoxy** job, select `0` instances.
    e. Click **Save**.

11. (Optional) If you are not using SSL encryption or if you are using self-signed certificates, you can select **Disable SSL certificate verification for this**

environment. Selecting this checkbox also disables SSL verification for route services.

> Use this checkbox only for development and testing environments. Do not select it for production environments.

12. (Optional) If you do not want HAProxy or the Gorouter to accept any non-encrypted HTTP traffic, select the **Disable HTTP on HAProxy and Router** checkbox.

13. Click **Save**.

14. After you complete the configuration in PCF, add your certificate or certificates to your load balancer, and configure its listening port. The procedures vary depending on your IaaS.

15. Configure your load balancer to append the `X-Forwarded-For` and `X-Forwarded-Proto` headers to client requests.

    If you cannot configure the load balancer to provide the `X-Forwarded-For` header, the Gorouter appends it in requests forwarded to applications and system components, set to the IP address of the load balancer.

    > **Note**: If the load balancer accepts unencrypted requests, it **must** provide the `X-Forwarded-Proto` header. Conversely, if the load balancer cannot be configured to send the `X-Forwarded-Proto` header, it should not accept unencrypted requests. Otherwise, applications and platform system components that require encrypted client requests will accept unencrypted requests when they should not accept them.

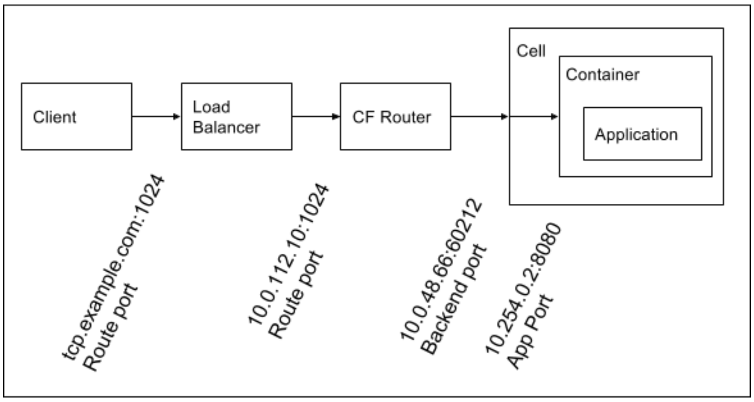
Enabling TCP Routing

Page last updated:

This topic describes enabling TCP Routing for your Cloud Foundry (CF) deployment. This feature enables developers to run applications that serve requests on non-HTTP TCP protocols. You can use TCP Routing to comply with regulatory requirements that require your organization to terminate the TLS as close to your apps as possible so that packets are not decrypted before reaching the application level.

Route Ports

The diagram below shows the layers of network address translation that occur in Cloud Foundry in support of TCP Routing. The descriptions step through an example work flow that covers route ports, backend ports, and app ports.



- A developer creates a TCP route for their application based on a TCP domain and a route port, and maps this route to one or more applications. See the Creating Routes topic for more information.
- Clients make requests to the route. DNS resolves the domain name to the load balancer.
- The load balancer listens on the port and forwards requests for the domain to the TCP routers. The load balancer must listen on a range of ports to support multiple TCP route creation. Additionally, Cloud Foundry must be configured with this range, so that the platform knows what ports can be reserved when developers create TCP routes.
- The TCP router can be dynamically configured to listen on the port when the route is mapped to an application. The domain the request was originally sent to is no longer relevant to the routing of the request to the application. The TCP router keeps a dynamically updated record of the backends for each route port. The backends represent instances of an application mapped to the route. The TCP Router chooses a backend using a round-robin load balancing algorithm for each new TCP connection from a client. As the TCP Router is protocol agnostic, it does not recognize individual requests, only TCP connections. All client requests transit the same connection to the selected backend until the client or backend closes the connection. Each subsequent connection triggers the selection of a backend.
- Because containers each have their own private network, the TCP router does not have direct access to application containers. When a container is created for an application instance, a port on the Cell VM is randomly chosen and iptables are configured to forward requests for this port to the internal interface on the container. The TCP router then receives a mapping of the route port to the Cell IP and port.
- The Diego Cell only routes requests to port `8080`, the App Port, on the container internal interface. The App Port is the port on which applications must listen.

Pre-Deployment Steps

Before enabling TCP Routing, you must complete the following steps to set up networking requirements.

1. Choose a domain name from which your developers will create TCP routes for their applications. For example, create a domain which is similar to your app domain but prefixed by the TCP subdomain: `tcp.APPS-DOMAIN.com`
2. Configure DNS to resolve this domain name to the IP address of a highly-available load balancer that will forward traffic for the domain to the TCP routers. For more information, view the Domains topic. If you are operating an environment that does not require high-availability, configure DNS to resolve the TCP domain name you have chosen directly to a single instance of the TCP Router.
3. (Optional) Choose IP addresses for the TCP routers and configure your load balancer to forward requests for the domain you chose in the step above to these addresses. Skip this step if you have configured DNS to resolve the TCP domain name to an instance of the TCP Router. Review the Enable TCP Routing steps in the *Deploying PAS* topic for your IaaS to configure your IP addresses for your PCF deployment: AWS, OpenStack, or vSphere.

4. (Optional) Decide on the number of TCP routes you want to support. For each TCP route, you must reserve a port. Configure your load balancer to forward the range of ports to the TCP routers. Skip this step if you have configured DNS to resolve the TCP domain name to an instance of the TCP Router.

5. Review the "Enable TCP Routing" steps in the *Deploying Pivotal Application Service* topic for your IaaS to configure your ports for your PCF deployment: AWS, OpenStack, or vSphere.

Post-Deployment Steps

In the following steps you use the Cloud Foundry Command Line Interface (cf CLI) to add the TCP shared domain and configure organization quotas to grant developers the ability to create TCP routes. This requires an admin user account.

Configure CF with Your TCP Domain

After deployment, you must configure Cloud Foundry with the domain that you configured in the Pre-Deployment step above so that developers can create TCP routes from it.

1. Run `cf router-groups`. You should see `default-tcp` as a response.

2. Run `cf create-shared-domain` to create a shared domain and associate it with the router group.

```
$ cf create-shared-domain tcp.APPS-DOMAIN.com --router-group default-tcp
```

3. Run `cf domains`. Verify that next to your TCP domain, `TCP` appears under `type`.

Configure a Quota for TCP Routes

Since TCP route ports are a limited resource in some environments, quotas are configured to allow creation of zero TCP routes by default. After you deploy Cloud Foundry, you can increase the maximum number of TCP routes for all organizations or for particular organizations and spaces. Because you reserve a route port for each TCP route, the quota for this resource is managed with the cf CLI command option `--reserved-route-ports`. See the Creating and Modifying Quota Plans topic for more information.

If you have a default quota that applies to all organizations, you can update it to configure the number of route ports that can be reserved by each organization.

```
$ cf update-quota QUOTA —reserved-route-ports X
```

To create a new organization quota that can be allocated to particular organizations, provide the following required quota attributes in addition to the number of reserved route ports:

```
$ cf create-quota QUOTA —reserved-route-ports X
```

You can also create a quota that governs the number of route ports that can be created in a particular space.

```
$ cf create-space-quota QUOTA —reserved-route-ports X
```

Create a TCP Route

For instructions about creating a TCP Route, see the Create a TCP Route with a Port topic.

Router Groups

In Post-Deployment Steps, we describe that in order to create a domain from which to create TCP routes, it must be associated with the TCP Router Group. A router group is a cluster of identically configured routers. Router Groups were introduced as mechanism to support reservation of the same port for multiple TCP routes, thus increasing the capacity for TCP routes. However, only one router group is currently supported. In the Pre-Deployment Steps.

we describe how an admin user can configure the port range available for TCP routes in preparation for deployment.

Modify your TCP ports

After deployment, you can modify the range of ports available for TCP routes using `cf curl` commands, as demonstrated with the commands below. These commands require an admin user with the `routing.router_groups.read` and `routing.router_groups.write` scopes.

1. In a terminal window, run `cf curl /routing/v1/router_groups` to view the `reservable_ports`:

```
$ cf curl /routing/v1/router_groups
[
  {
    "guid": "9d1c1da9-ed63-45e8-45ee-256f8579455c",
    "name": "default-tcp",
    "type": "tcp",
    "reservable_ports": "60000-60098"
  }
]
```

2. Modify the `reservable_ports`:

```
$ cf curl /routing/v1/router_groups/f7392031-a488-4890-8835-c4a038a3bded -X PUT -d '{"reservable_ports":"1024-1199"}'
```

Supporting WebSockets

Page last updated:

This topic explains how Cloud Foundry (CF) uses WebSockets, why developers use WebSockets in their applications, and how operators can configure their load balancer to support WebSockets.

Operators who use a load balancer to distribute incoming traffic across CF router instances must configure their load balancer for WebSockets. Otherwise, the Loggregator system cannot stream application logs to developers, or application event data and system metrics to third-party aggregation services. Additionally, developers cannot use WebSockets in their applications.

Understand WebSockets

The WebSocket protocol provides full-duplex communication over a single TCP connection. Applications can use WebSockets to perform real-time data exchange between a client and a server more efficiently than HTTP.

CF uses WebSockets for the following metrics and logging purposes:

1. To stream all application event data and system metrics from the Doppler server instances to the Traffic Controller
2. To stream application logs from the Traffic Controller to developers using the Cloud Foundry Command Line Interface (cf CLI) or Apps Manager
3. To stream all application event data and system metrics from the Traffic Controller over the Firehose endpoint to external applications or services

For more information about these Loggregator components, see the Overview of the Loggregator System topic.

Configure Your Load Balancer for WebSockets

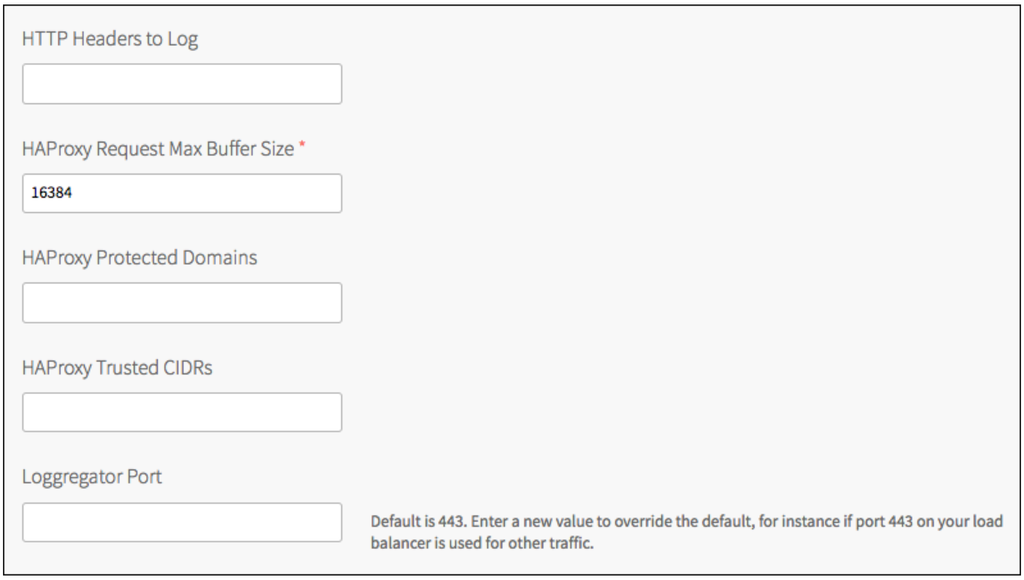
To form a WebSocket connection, the client sends an HTTP request that contains an `Upgrade` header and other headers required to complete the WebSocket handshake. You must configure your load balancer to not upgrade the HTTP request, but rather to pass the `Upgrade` header through to the CF router. The procedures required to configure your load balancer depends on your IaaS and load balancer. The following list includes several possible approaches:

- Some load balancers can recognize the `Upgrade` header and pass these requests through to the CF router without returning the WebSocket handshake response. This may or may not be default behavior, and may require additional configuration.
- Some load balancers do not support passing WebSocket handshake requests containing the `Upgrade` header to the CF router. For instance, the Amazon Web Services (AWS) Elastic Load Balancer (ELB) does not support this behavior. In this scenario, you must configure your load balancer to forward TCP traffic to your CF router to support WebSockets. If your load balancer does not support TCP pass-through of WebSocket requests on the same port as other HTTP requests, you can do one of the following:
  - Configure your load balancer to listen on a non-standard port (the built-in CF load balancer listens on 8443 by default for this purpose), and forward requests for this port in TCP mode. Application clients must make WebSockets upgrade requests to this port. If you choose this strategy, you must follow the steps below in the Set Your Loggregator Port section of this topic
  - If a non-standard port is not acceptable, add a load balancer that will handle WebSocket traffic (or another IP on an existing load balancer) and configure it to listen on standard ports 80 and 443 in TCP mode. Configure DNS with a new hostname, such as `ws.cf.example.com`, to be used for WebSockets. This hostname should resolve to the new load balancer interface.

Note: Regardless of your IaaS and configuration, you must configure your load balancer to send the X-Forwarded-For and X-Forwarded-Proto headers for non-WebSocket HTTP requests on ports 80 and 443. See the Securing Traffic into Cloud Foundry topic for more information.

Set Your Loggregator Port

By default, PCF assigns port 443 for TCP/WebSocket communications. If you have configured your load balancer to use a port other than 443 for TCP/WebSocket traffic, you must edit the **Loggregator Port** field in the **Networking** pane of the PAS tile.


HTTP Headers to Log
HAProxy Request Max Buffer Size *
16384
HAProxy Protected Domains
HAProxy Trusted CIDRs
Loggregator Port
Default is 443. Enter a new value to override the default, for instance if port 443 on your load balancer is used for other traffic.

Pivotal

Configuring Load Balancer Healthchecks for Cloud Foundry Routers

Page last updated:

This topic describes how to configure load balancer healthchecks for Cloud Foundry (CF) routers to ensure that the load balancer only forwards requests to healthy router instances. You can also configure a healthcheck for your HAProxy if your deployment uses the HAProxy component.

In environments that require high availability, operators must configure their own redundant load balancer to forward traffic directly to the CF routers. In environments that do not require high availability, operators can skip the load balancer and configure DNS to resolve the CF domains directly to a single instance of a router.

Add a Healthcheck Endpoint for HAProxy

If you have deployed one or more instances of HAProxy between your infrastructure load balancer and Gorouters, configure your infrastructure load balancer to use the following HTTP healthcheck endpoint: `http://HAPROXY_IP:8080/health`.

The HAProxy is an optional component that provides some features that Gorouter does not and can be helpful for demonstrating horizontal scalability of the CF routers in environments where an infrastructure load balancer is not available.

Set the Healthy and Unhealthy Threshold Properties for the Gorouter

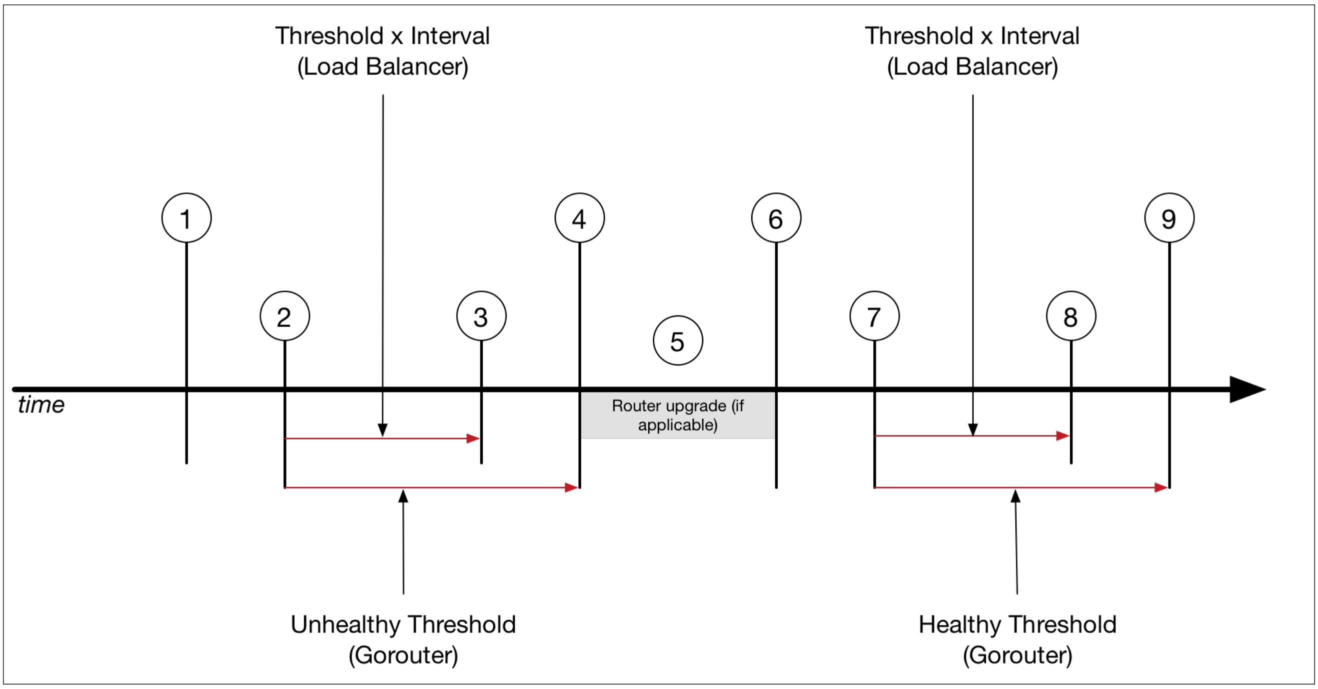
To maintain high availability during upgrades to the HTTP router, each router is upgraded on a rolling basis. During upgrade of a highly available environment with multiple routers, each router is shutdown, upgraded, and restarted before the next router is upgraded. This ensures that any pending HTTP request passed to the HTTP router are handled correctly.

Pivotal Application Service (PAS) uses the following properties:

- **Unhealthy Threshold**: Specifies the amount of time, in seconds, that the Router continues to accept connections before shutting down. During this period, the healthcheck reports `unhealthy` to cause load balancers to fail over to other routers. You should set this value greater than or equal to the maximum amount of time it could take your load balancer to consider a router instance unhealthy, given contiguous failed healthchecks.
- **Healthy Threshold**: Specifies the amount of time, in seconds, to wait until declaring the router instance started. This allows an external load balancer time to register the instance as `healthy`.

You can configure these properties from the **Settings** > **Network** tab.

The image and table below describe the behavior of the load balancer health checks when a router shuts down and is restarted.



Step	Description
1	A shutdown request is sent to the router.
2	The router receives shutdown request, which causes the following: • The router begins sending Service Unavailable responses to the load balancer health checks. • The load balancer continues sending HTTP request to the router
3	The load balancer considers the router to be in an unhealthy state, which causes the load balancer to stop sending HTTP requests to the router. The time between step 2 and 3 is defined by the values of the health check interval and threshold configured on the load balancer.
4	The router shuts down. The interval between step 2 and 4 is defined by the Unhealthy Threshold property of the Gorouter. In general, the value of this property should be longer than the value of the interval and threshold values (interval x threshold) of the load balancer. This additional interval ensures that any remaining HTTP requests are handled before the router shuts down.
5	If the router shutdown is initiated by an upgrade, the Gorouter software is upgraded.
6	The router restarts.
7	The routers begins returning Service Available responses to the load balancer health check.
8	The load balancer considers the router to be in a healthy state. The time between step 7 and 8 is specified by the health check interval and threshold configured for your load balancer (health check threshold x health check interval).
9	Shutdown and upgrade of the other router begins.

Pivotal

Router Backend Keepalive Connections

Page last updated:

This topic describes how to enable support in the Gorouter for keepalive connections with backends, and considerations for configuration.

Support for keepalive connections is described in Idle Keepalive Connections.

Enable Keepalive Connections from the Gorouter to Backends

By default, support for keepalive connections with backends is disabled. An operator can enable the feature by entering a non-zero value for the **Router Max Idle Keepalive Connections** field in the Networking Pane of the PAS tile.

Considerations for Configuring Max Idle Keepalive Connections

Each router process is limited to 100,000 file descriptors. Each inbound request consumes at most two file descriptors; one for the connection from the client and one from the Gorouter to the backend. For this reason we recommend that Max Idle Connections is never set higher than 50,000. As a few other file descriptors are consumed by the process under normal operation, this limit is more like 49,900.

In order to determine how many idle connections are needed, consider that the peak connections could be calculated by multiplying throughput by response time. Example: if a single router receives 1000 requests per second, and response time is 1 second, at any given time there are likely to be 1000 connections open to backends. Accommodating for growth of 2x, an operator might configure a maximum of 2000 idle connections per router.

The Gorouter has been hard coded with a limit of 100 idle connections per backend. The configurable property `max_idle_connections` governs idle connections across all backends.

Gorouter Retry Behavior

When you deploy an app that requires Diego cells to restart or recreate, the app may not respond to a Gorouter request before the keepalive connection breaks. The following table describes how the Gorouter behaves if it cannot establish a TCP connection to an app:

If the Gorouter …	then the Gorouter …
cannot establish a TCP connection to the routing backend for the app	waits 15 minutes for a response.
establishes a TCP connection to the routing backend, but the app does not respond	retries the request three times.

In both cases, if the app still does not respond to the request, the Gorouter returns a 502.

Using PAS for Windows 2012R2

Note: Elastic Runtime has been renamed Pivotal Application Service.

This documentation describes how operators install and manage Windows Diego cells in Pivotal Cloud Foundry (PCF) with PAS for Windows 2012R2 and how developers push .NET Framework applications to Windows cells.

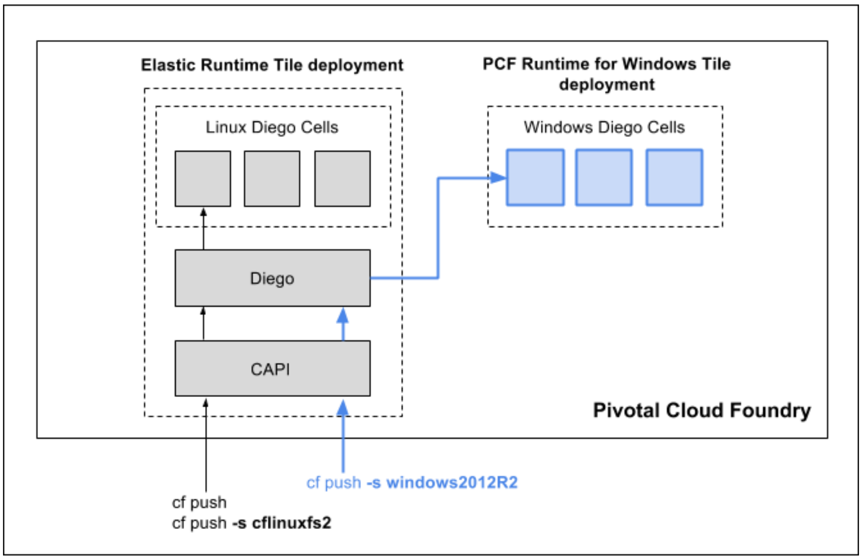
Operators can install PAS for Windows 2012R2 on Microsoft Azure, VMware vSphere, Amazon Web Services (AWS), or Google Cloud Platform (GCP).

Overview

Operators who want to enable developers to push .NET Framework applications can deploy Windows-based Diego cells in PCF with the PAS for Windows 2012R2 tile.

Deploying this tile creates a separate BOSH deployment populated with the Garden-Windows BOSH release, which runs on a Windows cell built from a Windows Server 2012R2 stemcell. This lets PCF deploy Windows-based Diego cells in addition to Linux-based Diego cells.

Once the Windows cell is running and after developers build their applications, they can then specify the `windows2012R2` stack when pushing .NET Framework apps from the command line or PowerShell session. Using the Diego infrastructure, PCF passes the app to the Windows cell in the PAS for Windows 2012R2 deployment. The diagram below illustrates the process.



Requirements

When configuring the Windows deployment, the minimum resource requirements for each Windows cell are as follows:

- Disk size: 64 GB
- Memory: 16 GB
- Number of CPUs: 4

By design, PAS for Windows 2012R2 only supports deployments based on Windows Server 2012R2.

Due to Microsoft's licensing requirements, operators must either bring their own licensed copies of Windows Server (for on-premise deployments) or pay the surcharges associated with Windows Server licensing determined by the IaaS provider on Azure, AWS, and GCP.

Understanding PAS for Windows 2012R2

- Understanding Windows Cells

- Understanding Stemcell Security

Installing PAS for Windows 2012R2

- Using Windows Stemcells
- Deploying PAS for Windows 2012R2

Managing Windows Cells

- Upgrading Windows Cells
- Configuring a KMS Host
- Troubleshooting Windows Cells

Developing on Windows Cells

- Deploying .NET Framework Apps to Windows Cells

Limitations

Deployments of Windows Diego Cells with PAS for Windows 2012R2 have the following limitations:

- Because of the process management characteristics of Windows Server 2012R2, Windows cells can host a maximum of 40 app instances per cell. Operators should take this density constraint into account when planning the sizes of their PAS for Windows 2012R2 deployments, in addition to CPU, disk, and memory needs.
- The following Cloud Foundry features are not supported by PAS for Windows 2012R2:
  - Diego SSH, i.e. the `cf ssh` command. This is due to limitations of the inherent file system isolation characteristics of Windows Server 2012R2.
  - Volume services. For mounting SMB shares, developers should access SMB volumes from their applications directly.
  - Container-to-container networking. The IronFrame library that provides container-like features on Windows Server 2012R2 does not support container networking. IronFrame binds virtual container ports to the host.
- The following Windows technologies are not supported by PAS for Windows 2012R2:
  - Active Directory Domain Services, i.e. joining Windows cells to an Active Directory domain.
  - Integrated Windows Authentication. Instead, operators should deploy Active Directory Federation Services and the Pivotal SSO tile to enable OAuth-based authentication.
- You cannot push Docker or other OCI-compatible images to Windows cells.

Known Issues

PAS for Windows 2012R2 has the following known issues:

- In the PAS for Windows 2012R2 tile **Credentials** tab, the `vcap` credentials that appear when you click **VM Credentials** > **Link to Credential** do not apply to Windows cells, since they do not yet have a vcap user.
- Under **VM Options** > **Manage Administrator Password** > **Set the password**, setting the `Administrator` user password has the following issues:
  - For Azure-hosted deployments: Setting the password directly does not yet work. All passwords for the user called `Administrator` will be randomized by default. Please use the following workarounds to access the VM directly:
    - Create a user in the Azure cloud management console.
    - Use the Enable SSH feature.
  - For GCP-hosted deployments: Setting the password directly does not yet work. Please use the following workarounds to access the VM:
    - Create a user in the GCP cloud management console.
    - Use the Enable SSH feature.

    - For AWS and vSphere deployments: You can set a password for the user called `Administrator`, but the password must be 8-14 characters and is limited to letters, numbers, and the `!` character, which is the only working special character.
    - For all IaaSes, for stemcell versions 1200.5 and later, the password for the user called `Administrator` is randomized by default to provide additional security in production settings.

- The controls in the **Advanced Features** pane have no effect.
- In the **Resource Config** pane, setting VM disk sizes has the following limitations. See the Root Disk Sizing table for details:
    - AWS: For stemcells earlier than 1200.9, setting disk size value has no effect. VM disks are 30 GB.
    - GCP: Setting the disk size only works for values 50 GB or larger.
    - vSphere: Setting disk size value has no effect. VM disks will match the disk size of the stemcell you create.

Understanding Windows Cells

This topic provides a description of how Windows cells work in Pivotal Cloud Foundry (PCF). For more information about security, see the Understanding Stemcell Security topic.

Overview

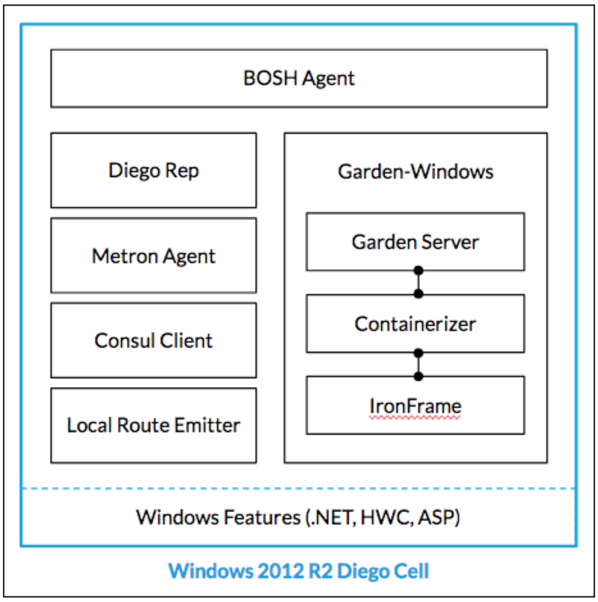
App instances in PCF run inside containers. Garden is the API that creates and manages these containers, and Garden Windows implements Garden on Windows.

By deploying the PAS for Windows 2012R2 tile, operators create a Windows cell from a stemcell that contains the Windows Server 2012 operating system. Because Windows does not natively support Linux-style containers, Garden Windows uses an open-source library called IronFrame to implement containerization on Windows. IronFrame uses features of the Windows kernel to isolate resources that would otherwise be shared, creating containers comparable to those that exist on Linux.

A Windows cell includes the following components:

- Garden Windows: Implements the Garden API on Windows
- Containerizer: Creates and manages Windows containers, using the IronFrame library
- Metron Agent: Forwards app logs, errors, and metrics to the Loggregator system
- BOSH Agent: Executes instructions from the BOSH Director
- Consul Client: Registers the cell as a service in a Consul cluster
- Cell Rep: Runs and manages Tasks and Long Running Processes

The following diagram illustrates the architecture of a Windows cell:



Garden Windows achieves container isolation in the following ways:

- Filesystem Isolation
- Disk Usage
- Network Isolation
- Memory Usage

Filesystem Isolation

Garden Windows creates a unique temporary user for each container, and uses Access Control Lists (ACLs) to render a “containerized” directory visible only to the user who owns the container. The temporary user can also read much of the host cell's filesystem, such as system DLLs and `C:\Program Files`.

Note: Because the temporary user who owns the container can view much of the host filesystem, operators should avoid placing sensitive or confidential information in system directories that would be accessible by standard users on a Windows workstation.

Disk Usage

Garden Windows enforces disk usage limits with NTFS disk quotas, which work on a per-user, per-volume basis. The disk quotas apply to the temporary user who owns the container, on the volume that contains the containerized directory, which is `C:\` by default. Because quotas are transparent to the user, the temporary user who owns the container can only see the disk resources available within the assigned quota.

Network Isolation

Apps launched inside a Garden Windows container bind directly to the external IP address of the cell. For apps that utilize the port mapping functions of the Garden API, Garden Windows maps internal container ports to external ports.

Memory Usage

Garden Windows uses job objects to enforce limits on memory usage by an app inside a container. A job object is a Windows kernel object that enables the control of multiple processes as a single group. Garden Windows assigns the processes inside a container to a job object and sets an upper limit on memory utilization by this job object, which is enforced by the kernel.

Additionally, an IronFrame component called the Guard helps enforce memory limits. The Guard polls for app memory usage and ensures that no app has mapped more memory than allowed. If an app exceeds its memory limit, the Guard kills the job object. If a process attempts to escape a job object, the Guard stops this behavior and kills the process if necessary.

Windows Stemcell Hardening

This topic provides a description of the security measures that Pivotal uses to harden the Windows stemcell.

Local Group Policy Settings

The Windows stemcell contains a version of Windows Server 2012 R2 with a set of Local Group Policy settings optimized for security. These settings begin with the WS2012R2 Member Server Security Compliance v1.0 baseline, included in Microsoft Security Compliance Manager v4.0. For more information about this baseline, see the Microsoft Security Guidance blog.

Pivotal has collaborated with Microsoft to further harden the stemcell by implementing Local Security Policies settings, according to the recommended security baseline defined in Microsoft Security Compliance Manager. The table below lists these overrides.

Note: Pivotal will continue to revise these settings as Microsoft releases updates.

Name	Setting
Turn off Automatic Download and Install of updates	Enabled
Allow Remote Shell Access	Disabled
Windows Firewall: Private: Display a notification	No
Windows Firewall: Domain: Display a notification	No
Windows Firewall: Public: Display a notification	No
Network access: Do not allow storage of passwords and credentials for network auth	Enabled
Access this computer from the network	Administrators
Deny log on as a batch job	Guests
Deny log on as a service	Guests
Deny log on through Remote Desktop Services	Guests

Installing PAS for Windows 2012R2

This topic provides links to additional information about deploying PAS for Windows 2012R2:

- Using Windows Stemcells
- Deploying PAS for Windows 2012R2

Using Windows Stemcells

Overview

A *stemcell* is a customized operating system image containing the filesystem for BOSH-managed virtual machines. When deployed, it includes a process, the *BOSH Agent*, which is pre-configured to communicate with the orchestrating VM, the *BOSH Director*. The BOSH Agent is responsible for executing and monitoring jobs on the VM.

PAS for Windows 2012R2 deployments currently use a stemcell based on Windows Server 2012 R2.

Root Disk Sizing

The size of the stemcell's root disk determines the minimum root disk size of any Windows VM that the BOSH Director can create from that stemcell. The relationship between a stemcell's disk size and the root disk size of the Windows cells depends on your IaaS as shown in the table below.

IaaS	Stemcell root disk size	Possible root disk sizes for Windows cells
Azure	127 GB	127 GB
AWS	30 GB	30 GB or larger (for stemcell 1200.9 and later)
GCP	50 GB	50 GB or larger
vSphere	Recommended 128 GB+*	Same as stemcell minimum root disk size

*Because you create your own stemcell on vSphere, you can control its root disk size. See Creating a vSphere Stemcell Manually for more information.

In the PAS for Windows 2012R2 tile > **Resource Config** pane, the VM types that you can choose from reflect the disk size constraints in the table above.

Light Stemcells

Light stemcells are metadata files that enable you to deploy VMs without uploading the full operating system manually. These stemcells reference images of Windows Server 2012 R2 hosted on public IaaSes.

Light stemcells are available for Microsoft Azure, AWS, and GCP, which typically bill a surcharge for the associated Windows licensing cost. See the documentation on those IaaSes for details.

Below is IaaS-specific documentation for light stemcells you can use with the PAS for Windows 2012R2 tile:

- Deploying the Azure Light Stemcell

Stemcell Builds

Because of the constraints placed on the distribution of Windows OS by Microsoft's licensing terms, operators must build stemcells for on-premise deployments from OS disk images (e.g. ISO files) licensed to them directly from Microsoft. This typically involves starting a Windows virtual machine, installing the BOSH agent and associated software, then exporting its disk image and converting it to a stemcell format.

Below is IaaS-specific documentation for building stemcells for the PAS for Windows 2012R2 tile:

- vSphere Stemcell Build

Deploying the Azure Light Stemcell

This topic explains how to download and use the Azure light stemcell. For more information about *light stemcells*, see the Light Stemcells section of the *Using Windows Stemcells* topic.

Download the Azure Light Stemcell

To download the Azure light stemcell, perform the following steps:

1. Navigate to Pivotal Network.
2. Select **Azure Light Stemcell for Windows 2012R2 Server** from the **Release Download Files** section of the Stemcells for PCF (Windows) page.

To start using the Azure light stemcell, perform the steps in the Enable the Azure Light Stemcell and Configure Resources sections.

Enable the Azure Light Stemcell

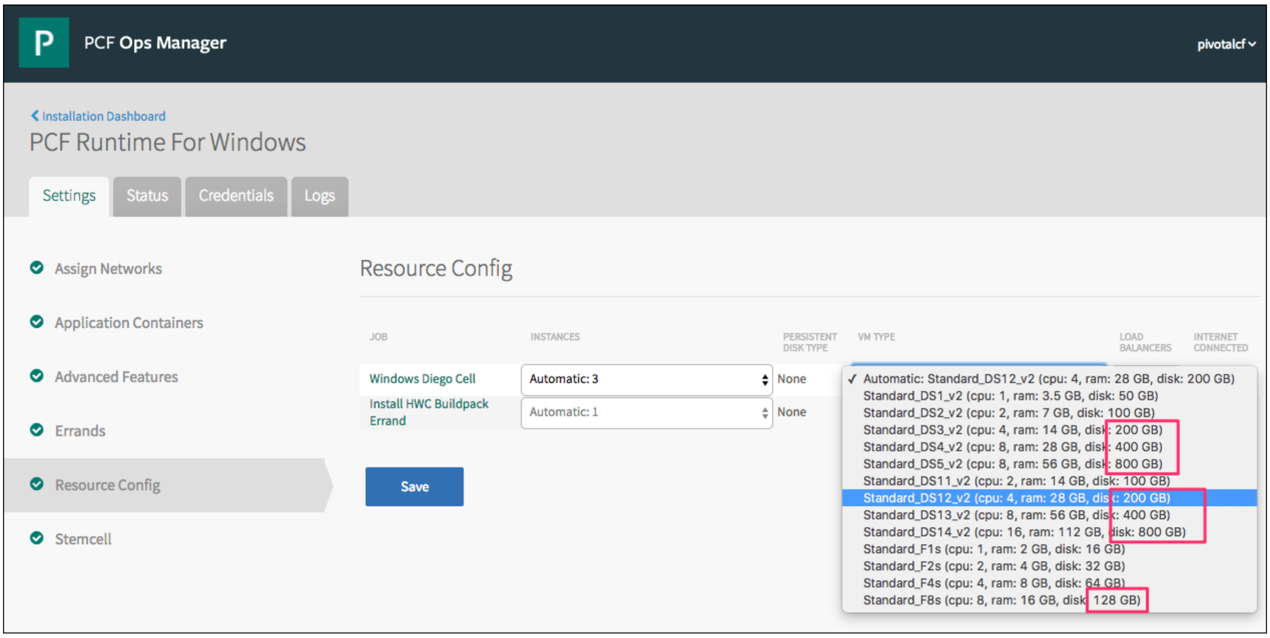
To use the Azure light stemcell, you must accept the corresponding license agreement and enable the stemcell for your non-trial Azure subscription through the Azure marketplace as follows:

1. Navigate to https://portal.azure.com and log in.
2. From the options on the left side of the page, click **+ New**.
3. In the **Search the marketplace** bar, search for **BOSH Azure Windows Stemcell**.
4. Select **BOSH Azure Windows Stemcell** from your search results. A description of the BOSH Azure Windows Stemcell appears.
5. Below the description, at the bottom of the page, click the blue banner that reads **Want to deploy programmatically? Get started ➔**.
6. Review the Terms of Use.
7. Under **Choose the subscriptions**, click **Enable** for each Azure subscription with which you want to use the stemcell.
8. Click **Save**.

Configure Resources

When you deploy or upgrade the PAS for Windows 2012R2 tile on Azure, configure the resources of Ops Manager Director and PAS for Windows 2012R2 as follows:

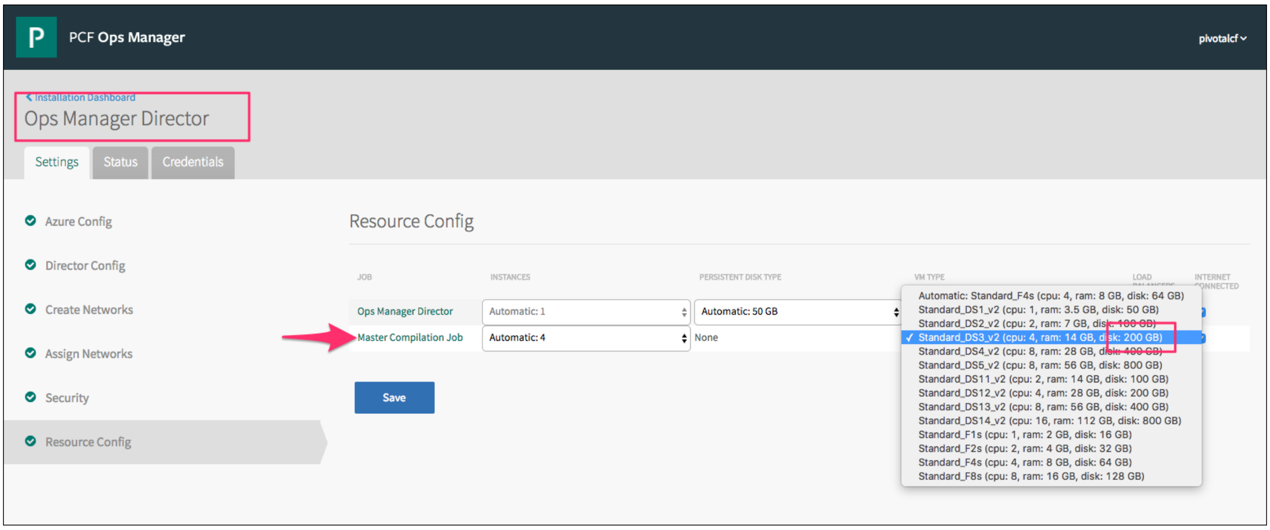
1. On the Ops Manager Installation Dashboard, select **PAS for Windows 2012R2**.
2. In the **Settings** tab, select **Resource Config**.
3. In the **VM Type** field, ensure that **Windows Diego Cell** has a disk size of 128 GB or larger.
4. Ensure that **Install HWC Buildpack Errand** has a VM disk size of 128 GB or larger.



5. Navigate to **Ops Manager Director**.

6. In the **Settings** tab, select **Resource Config**.

7. In the **VM Type** field, ensure that the Ops Manager compilation VM has an ephemeral disk of 128 GB or larger.



For information about how to deploy and configure the PAS for Windows 2012R2 tile, see Deploying PAS for Windows 2012R2.

Creating a vSphere Stemcell Manually

In vSphere, customers must build their own stemcells to use PAS for Windows 2012R2 in their on-premise deployments.

Create a stemcell by following the directions in Creating a vSphere Stemcell by Hand.

The process includes starting with an ISO of Windows Server 2012R2 and installing the latest Windows Updates on a new virtual machine. Then you install the BOSH dependencies. Finally, you use the `stembuild` tool to package the VM’s disk as a `.tgz` file that can be uploaded to the PAS for Windows 2012R2 tile in Ops Manager.

Deploying PAS for Windows 2012R2

This topic describes how to install and configure the Pivotal Application Service (PAS) for Windows 2012R2 tile. The PAS for Windows 2012R2 tile installs Windows cells on your Pivotal Cloud Foundry (PCF) deployment.

Requirements

To install the PAS for Windows 2012R2 v1.12 tile, you must have Ops Manager v1.12.0 or later and Elastic Runtime v1.12.0 or later deployed to vSphere, Amazon Web Services (AWS), Google Cloud Platform (GCP), or Azure.

Step 1: Confirm Elastic Runtime Settings

There are two settings in the Elastic Runtime tile that affect the Windows cells installed by the PAS for Windows 2012R2 tile. Configure these settings as desired:

- In the **Networking** section, if you select the **Disable SSL certificate verification for this environment** checkbox, SSL certificate verification is disabled for Windows cells.
- In the **System Logging** section, if you configure an external syslog aggregator, logs are drained from Windows cells as well as non-Windows cells.

Step 2: Install the Tile

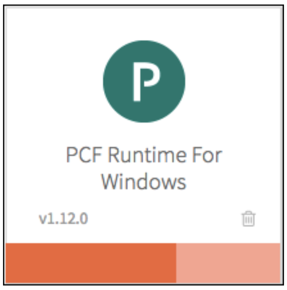
1. Download the product file from the **Pivotal Application Service for Windows 2012R2** section of Pivotal Network.

2. Navigate to the Ops Manager Installation Dashboard and click **Import a Product** to upload the product file.

3. Click **+** under the uploaded **PAS for Windows 2012R2** tile to add it to your staging area.

Step 3: Configure and Deploy the Tile

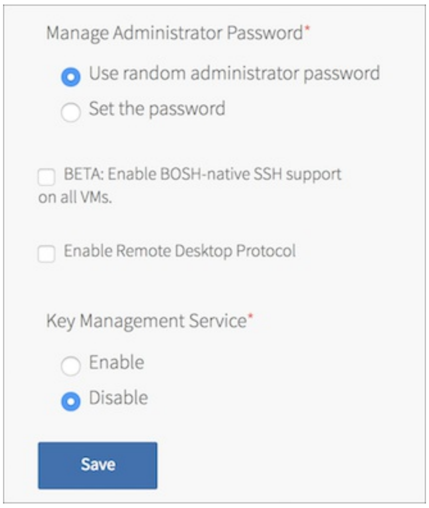
1. Click the newly added **PAS for Windows 2012R2** tile.



2. Click **Assign AZs and Networks** or **Assign Networks**. The name of the section varies depending on your IaaS.

3. Assign your AZs and networks and click **Save**.

4. Click **VM Options**.



5. Specify how you would like to manage the administrator password for your Windows VMs. The default option is to **Use random administrator password**.

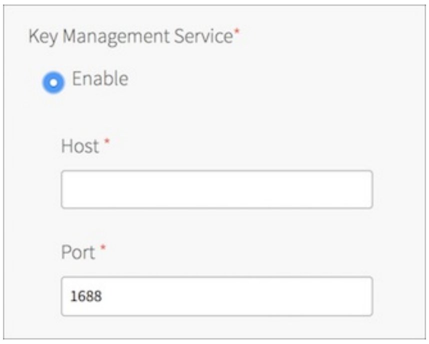
6. (Optional) Select the **BETA: Enable BOSH-native SSH support on all VMs** checkbox to start the Microsoft beta port of the OpenSSH daemon on port 22 on all VMs. Users will be able to SSH onto Windows VMs with the `bosh ssh` command, and enter a CMD terminal as an administrator user. They can then run `powershell.exe` to start a PowerShell session.

> **Note**: This feature is beta and not considered production-ready.

7. (Optional) If you want all VMs to support connection through Remote Desktop Protocol (RDP), click **Enable Remote Desktop Protocol**.

8. (Optional) If you want to configure a Key Management Service (KMS) that your volume-licensed Windows cell can register with, perform the following steps:

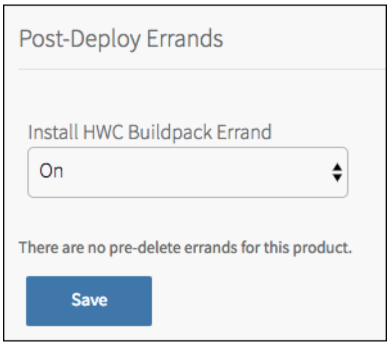
   a. Click **Enable**
   b. For the **Host** field, enter the KMS hostname.
   c. For the **Port** field, enter the port number. The default port number is `1688`.



9. (Optional) If you want your PAS for Windows 2012R2 application workloads to be deployed to an isolation segment, perform the steps in Assign a Tile to an Isolation Segment below.

10. (Optional) To configure Windows cells to send Windows Event logs to an external syslog server, click **System Logging** and perform the steps in the Send Cell Logs to a Syslog Server section.

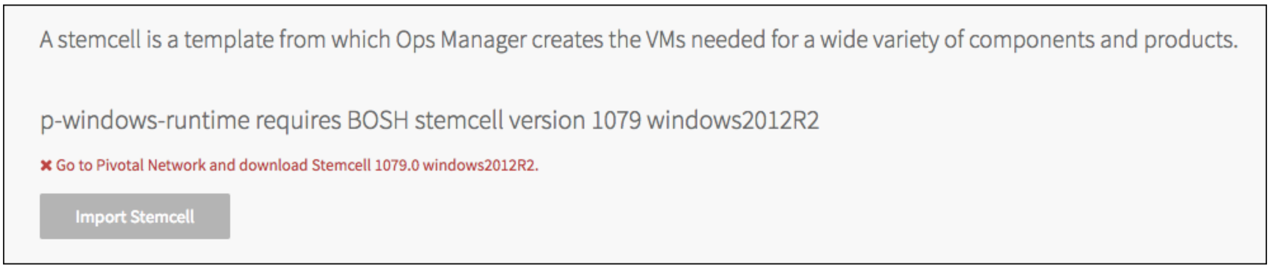
11. Click **Errands**. Pivotal recommends that you set the **Install HWC Buildpack Errand** to **On**. This ensures that you receive the most up-to-date HWC Buildpack.



12. Click **Resource Config** and adjust the number and VM types of your Windows cells as needed. Ensure that all VMs have disk sizes that match the possible sizes listed in the Root Disk Sizing table. If you are deploying PAS for Windows 2012R2 on Azure, see the Configure Resources section of the *Deploying the Azure Light Stemcell* topic.

13. Click **Save**.

14. Click **Stemcell**. Record the stemcell version number required by the tile.



15. Retrieve the stemcell by following the steps below for your IaaS.
    - For vSphere, you must create your own stemcell. For more information, see either Creating a vSphere Stemcell Manually or Creating a vSphere Stemcell by Hand.
    - For AWS, GCP, and Azure, navigate to the **Stemcells for PCF (Windows)** page of Pivotal Network. For more information, see the Light Stemcells section of the *Using Windows Stemcells* topic.

16. Click **Save**.

17. Return to the Ops Manager Installation Dashboard and click **Apply Changes** to install the PAS for Windows 2012R2 tile.

Step 4: (Optional) Create More Tiles

If you want to run Windows cells in multiple isolation segments, you must create and configure additional PAS for Windows 2012R2 tiles. See Windows Cells in Isolation Segments below for how to do this.

Windows Cells in Isolation Segments

Isolation segments are compartmentalized resource pools for Diego cells. The cells in one isolation segment share routing, computing, and logging resources with other cells in the same segment, and do not use resources from other isolation segments.

To run Windows cells in multiple isolation segments, you need to create and install multiple PAS for Windows 2012R2 tiles and configure each to run in a different isolation segment.

If the isolation segments do not already exist in the Cloud Controller Database (CCDB), you need to create them there.

See Replicate a Tile for how to create multiple copies of the PAS for Windows 2012R2 tile.

See Assign a Tile to an Isolation Segment for how to associate a PAS for Windows 2012R2 tile with an isolation segment, so that its cells run in that segment.

See the Create an Isolation Segment section of the *Managing Isolation Segments* topic for how to add an isolation segment to the CCDB.

Replicate a Tile

To make multiple copies of the **PAS for Windows 2012R2** tile that you can assign to different isolation segments, use the Replicator tool as follows:

1. Download the Replicator tool from the **PAS for Windows 2012R2** section of Pivotal Network.
2. Navigate to the directory where you downloaded the Replicator tool.
3. Replicate the tile:

```
./replicator \
  -name "YOUR-NAME" \
  -path /PATH/TO/ORIGINAL.pivotal \
  -output /PATH/TO/COPY.pivotal
```

   Replace the values above with the following:
   - `YOUR-NAME` : Provide a unique name for the new PAS for Windows tile. The name must be ten characters or less and only contain alphanumeric characters, dashes, underscores, and spaces.
   - `/PATH/TO/ORIGINAL` : Provide the absolute path to the original PAS for Windows tile you downloaded from Pivotal Network.
   - `/PATH/TO/COPY` : Provide the absolute path for the copy of the PAS for Windows tile that the Replicator tool produces.

4. Install and configure the Windows isolation segment, using the new `.pivotal` file and following the procedures in this topic, starting with the **Import a Product** step of Step 2: Install the Tile.

Assign a Tile to an Isolation Segment

To assign a PAS for Windows 2012R2 tile to an isolation segment, perform the following steps:

1. Open the **Application Containers** pane.

2. Under **Segment Name**, enter the name for the isolation segment to associate the tile with. If you are creating a new isolation segment, ensure that this name is unique across your PCF deployment.
3. Click **Save**.
4. If you are creating a new isolation segment, follow the steps in the Create an Isolation Segment section of the *Managing Isolation Segments* topic to add the isolation segment to the CCDB.

Upgrading Windows Cells

This topic describes how to upgrade and rotate the credentials for the PAS for Windows 2012R2 tile, update the Windows stemcell, and upgrade your Windows apps.

When to Upgrade

You must upgrade the PAS for Windows 2012R2 tile at the same time or immediately after you upgrade the PAS tile.

Unless otherwise noted, any minor version of PAS for Windows 2012R2 is compatible with the same minor version of PAS. For example, any 1.12.x version of PAS for Windows 2012R2 works with any 1.12.x version of PAS.

Upgrade PAS for Windows 2012R2

To upgrade PAS for Windows 2012R2:

1. Follow the instructions Upgrading PCF Products with the latest PAS for Windows 2012R2 product download from Pivotal Network.

2. If necessary, configure the product. For more information about configuring PAS for Windows 2012R2, see the Deploying PAS for Windows 2012R2 topic.

3. Locate the **Pending Changes** section on the right-hand side of the screen and click **Apply Changes**.

Rotate PAS for Windows 2012R2 Credentials

The PAS tile handles all of the credentials for PAS for Windows 2012R2.

To rotate your credentials in PAS for Windows 2012R2, re-deploy PAS and PAS for Windows 2012R2 by navigating to the Ops Manager Installation Dashboard and clicking **Apply Changes**.

Upgrade the Windows Stemcell

1. Navigate to the **Ops Manager Installation Dashboard**.

2. Retrieve the stemcell by following the steps below for your IaaS:
    - For vSphere, you must build your own stemcell.
    - For AWS, GCP, and Azure, navigate to the **Stemcells for PCF (Windows Server)** section of Pivotal Network.

3. Click the **PAS for Windows 2012R2** tile.

4. Within the **Settings** tab, click **Stemcell** on the left-hand navigation.

5. Click **Import Stemcell**.

6. Select the previously acquired stemcell and click **Open**.

7. Click **Installation Dashboard** to navigate back to the **Ops Manager Installation Dashboard**.

8. Click **Apply Changes**.

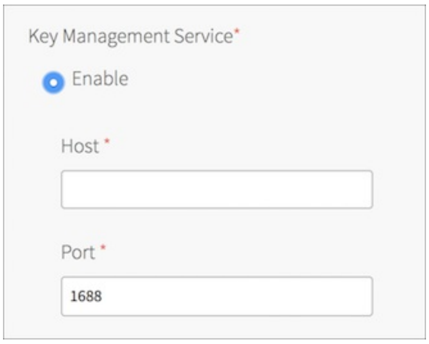
For more information about Windows stemcells, see Using Windows Stemcells.

Configuring a KMS Host

This topic describes how to configure a KMS host that your volume-licensed Windows cell can register with.

Configure a KMS Host

1. Navigate to the **PAS for Windows 2012R2** tile in Ops Manager
2. Select the **VM Options** configuration pane.
3. Under **Key Management Service**, perform the following steps:
   a. Click **Enable**
   b. For the **Host** field, enter the KMS hostname.
   c. For the **Port** field, enter the port number. The default port number is `1688`.



Troubleshooting Windows Cells

This topic describes how to troubleshoot Windows cells deployed by Pivotal Application Service (PAS) for Windows 2012R2.

Windows Cell Log Types

Windows cells generate two types of logs:

- **BOSH job logs**, such as `rep_windows` and `consul_agent_windows`. These logs stream to the syslog server configured in the PAS tile **System Logging** pane, along with other PCF component logs. The names of these BOSH job logs correspond to the names of the logs emitted by Linux Diego cells.
- **Windows Event logs**. These stream to the syslog server configured in the PAS for Windows 2012R2 **System Logging** pane and are downloadable through Ops Manager, as described below.

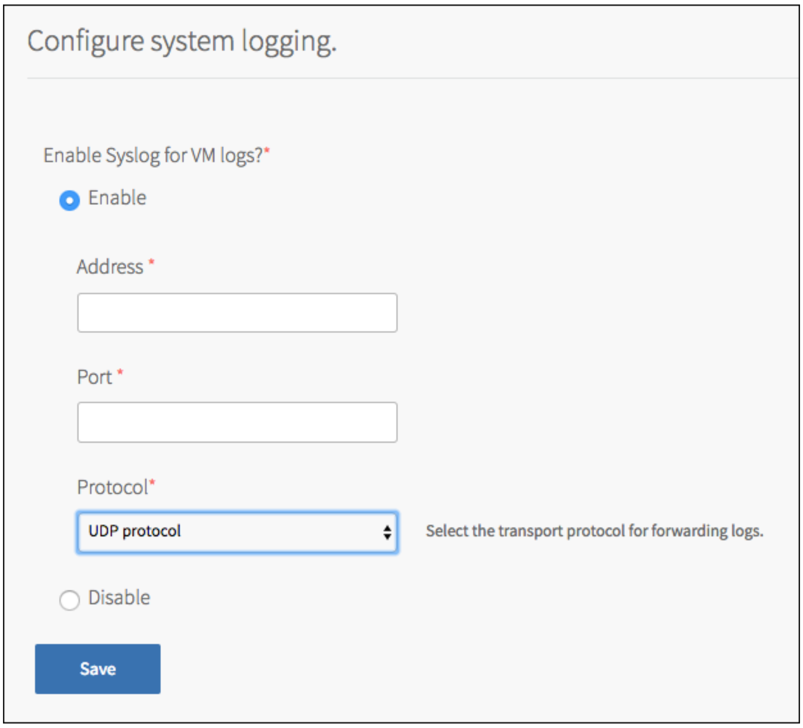
Access Windows Event Logs

PCF operators can access log messages from Windows Diego cells in two ways:

- Configure PAS for Windows to send all Windows cell logs to an external syslog server.
- Download archived logs from each Windows cell individually.

Send Cell Logs to a Syslog Server

To forward Windows cell log messages to an external syslog server, complete the following steps:

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the **PAS for Windows 2012R2** tile.
3. Under the **Settings** tab, select the **System Logging** pane.


4. Under **Enable Syslog for VM logs?**, click **Enable**

5. Under **Address**, enter the IP address of your syslog server.

6. Under **Port**, enter the port of your syslog server. The typical port for a syslog server is `514`.

    Note: The host must be reachable from the PAS network. Ensure your syslog server listens on external interfaces.

7. Under **Protocol**, select the transport protocol to use when forwarding logs.

8. Click **Save**.

Download Cell Logs

Perform the following steps to retrieve the logs for the Windows cell:

1. Navigate to the Ops Manager Installation Dashboard.

2. Click the **PAS for Windows 2012R2** tile.

3. Click the **Status** tab.

4. Under the **Logs** column, click the download icon for the Windows cell you want to retrieve logs from.

5. Click the **Logs** tab.

6. When the logs are ready, click the filename to download them.

7. Unzip the file to examine the contents. Each component on the cell has its own logs directory:
    - `/consul_agent_windows/`
    - `/garden-windows/`
    - `/metron_agent_windows/`
    - `/rep_windows/`

Connect to the Windows Cell

Perform the following steps to connect to your Windows cell to run diagnostics:

1. Download and install a Remote Desktop Protocol (RDP) client.
    - For Mac OS X, download the Microsoft Remote Desktop app from the Mac App Store.
    - For Windows, download the Microsoft Remote Desktop app from Microsoft.
    - For Linux/UNIX, download an RDP client like rdesktop.

2. Upload the runtime config. **Note:** The runtime config applies to all VMS managed by BOSH, and therefore will enable RDP on all Windows VMs.

```
$ bosh update runtime-config <runtime config filename>
```

Run the following command to retrieve the IP address of the Windows cell you would like to connect to:

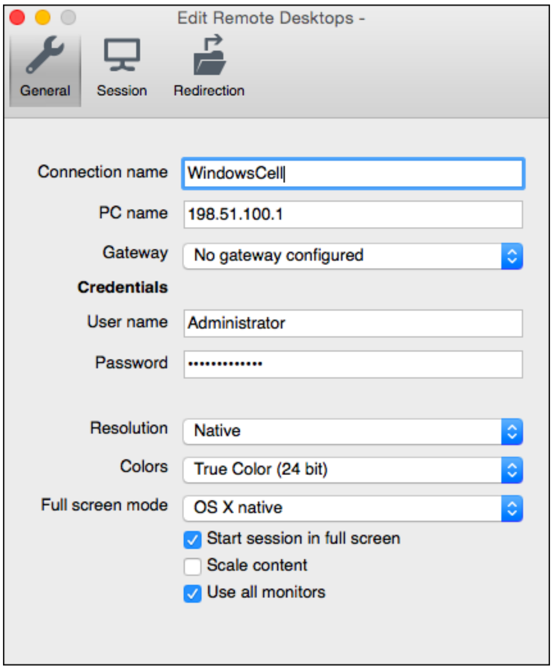
```
$ bosh vms
```

Obtain login credentials for your Windows cell.

- AWS or vSphere: if you chose **Set the password** under **VM Options** > **Manage Administrator Password** when you configured the PCF Runtime for Windows tile, the username is `Administrator` and the password is the one you entered. If you did not set the password, it is not possible to obtain the current Administrator password through the IaaS, because the BOSH agent randomized the password.
- GCP: follow the linked instructions for obtaining credentials.
- Azure: follow the linked instructions for obtaining credentials.

Establish an RDP connection

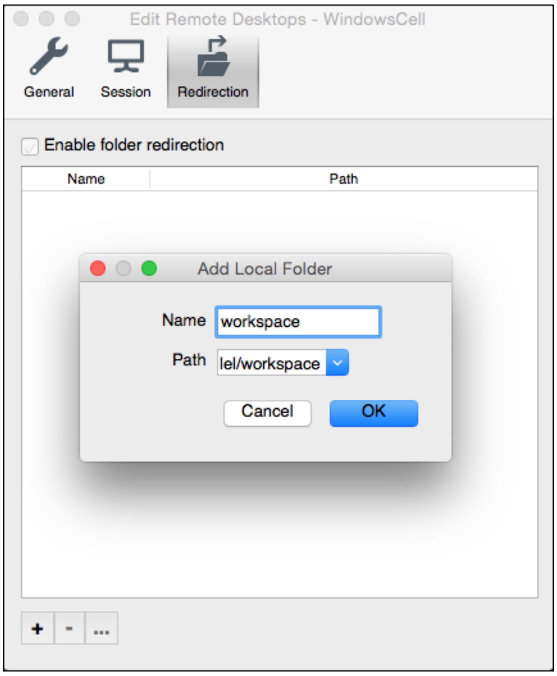
- Open your RDP client. The examples below use the Microsoft Remote Desktop app.



- Click **New** and enter your connection information:
  - **Connection name**: Enter a name for this connection.
  - **PC name**: Enter the IP address of your Windows cell.
  - **User name**: Enter the appropriate username.
  - **Password**: Enter the password of your Windows cell that you created.

1. To mount a directory on your local machine as a drive in the Windows cell, perform the following steps:

   a. From the same **Edit Remote Desktops** window as above, click **Redirection**.



   b. Click the plus icon at the bottom left.
   c. For **Name**, enter the name of the drive as it will appear in the Windows cell. For **Path**, enter the path of the local directory.
   d. Click **OK**.

2. Close the **Edit Remote Desktops** window and double-click the newly added connection under **My Desktops** to open a RDP connection to the Windows cell.

3. In the RDP session, you can use the Consul CLI to diagnose problems with your Windows cell.

Notes:

- **For vSphere**, it is also possible to connect to your windows cell using the vCenter console once the Administrator password has been set.
- **If the Windows is not accessible from your host**, tunnel through the Ops Manager VM. To do this, SSH into the Ops Manager VM with the flag `-L 3389:<Windows Cell IP>:3389` and then use IP 127.0.0.1 from your RDP client.

Consul CLI

Perform the following steps to use the Consul CLI on your Windows cell to diagnose problems with your Consul cluster:

1. In your RDP session, open a PowerShell window.
2. Change into the directory that contains the Consul CLI binary:

```
PS C:\Users\Administrator> cd C:\var\vcap\packages\consul-windows\bin\
```

3. Use the Consul CLI to list the members of your Consul cluster:

```
PS C:\Users\Administrator\var\vcap\packages\consul-windows\bin> .\consul.exe members
Node                    Address          Status  Type   Build  Protocol  DC
cell-windows-0          10.0.0.111:8301  alive   client  0.6.4  2        dc1
cloud-controller-0      10.0.0.94:8301   alive   client  0.6.4  2        dc1
cloud-controller-worker-0  10.0.0.99:8301   alive   client  0.6.4  2        dc1
consul-server-0         10.0.0.96:8301   alive   server  0.6.4  2        dc1
diego-brain-0           10.0.0.109:8301  alive   client  0.6.4  2        dc1
diego-cell-0            10.0.0.103:8301  alive   client  0.6.4  2        dc1
diego-cell-1            10.0.0.104:8301  alive   client  0.6.4  2        dc1
diego-cell-2            10.0.0.107:8301  alive   client  0.6.4  2        dc1
diego-database-0        10.0.0.92:8301   alive   client  0.6.4  2        dc1
ha-proxy-0              10.0.0.254:8301  alive   client  0.6.4  2        dc1
nfs-server-0            10.0.0.100:8301  alive   client  0.6.4  2        dc1
router-0                10.0.0.105:8301  alive   client  0.6.4  2        dc1
uaa-0                   10.0.0.93:8301   alive   client  0.6.4  2        dc1
```

4. Examine the output to ensure that the `cell-windows-0` service is registered in the Consul cluster and is `alive`. Otherwise, your Windows cell cannot communicate with your PCF deployment and developers cannot push .NET apps to the Windows cell. Check the configuration of your Consul cluster, and ensure that your certificates are not missing or misconfigured.

Deploying .NET Apps to Windows Cells

This topic describes how to push .NET apps to Windows cells.

Overview

After operators install and configure the PAS for Windows 2012R2 tile, developers can push .NET apps to the Windows cell. For documentation about installing the PAS for Windows 2012R2 tile, see the Deploying PAS for Windows 2012R2 topic. Developers can push both OWIN and non-OWIN apps, and can push apps that are served by Hostable Web Core or self-hosted.

Operators must also install the Cloud Foundry Command Line Interface (cf CLI) to run the commands on this topic. For information about installing the cf CLI, see the Installing the cf CLI topic.

If you have upgraded to PAS for Windows 2012R2 1.11 and have apps that you want to migrate, see the Upgrading Windows Cells topic.

Push a .NET App

By default, PCF serves .NET apps with Hostable Web Core (HWC). HWC is a lighter version of the Internet Information Services (IIS) server that contains the core IIS functionality.

Perform the following steps to push a .NET app to a Windows cell:

1. Run `cf api api.YOUR-SYSTEM-DOMAIN` to target the Cloud Controller of your PCF deployment:

   ```
   $ cf api api.YOUR-SYSTEM-DOMAIN
   ```

2. Run one of the following commands to deploy your .NET app, replacing `APP-NAME` with the name of your app.

   - If you want to deploy a .NET Framework app, run `cf push APP-NAME -s windows2012R2 -b hwc_buildpack`:

     ```
     $ cf push APP-NAME -s windows2012R2 -b hwc_buildpack
     ```

   - If you want to deploy an app with `.bat` or `.exe` files, run `cf push -s windows2012R2 -b binary_buildpack`:

     ```
     $ cf push -s windows2012R2 -b binary_buildpack
     ```

   The `-s windows2012R2` option instructs PCF to run the app in the Windows cell. If you are not pushing your app from its directory, use the `-p` option and specify the path to the directory that contains the app.

3. Wait for your app to stage and start. If you see an error message, refer to the Troubleshoot App Errors section of this topic.

Push a Self-Hosted App

Developers can choose to push a self-hosted app instead of using Hostable Web Core. Self-hosted apps combine server code with the app code.

Perform the following steps to push a self-hosted app:

1. Run `cf api api.YOUR-SYSTEM-DOMAIN` to target the Cloud Controller of your PCF deployment:

   ```
   $ cf api api.YOUR-SYSTEM-DOMAIN
   ```

2. Run `cf push APP-NAME -s windows2012R2 -b binary_buildpack -c PATH-TO-BINARY` to push your .NET app from the app root. Replace `APP-NAME` with the name of your app and `PATH-TO-BINARY` with the path to your executable.

   ```
   $ cf push APP-NAME -s windows2012R2 -b binary_buildpack -c PATH-TO-BINARY
   ```

3. Wait for your app to stage and start. If you see an error message, refer to the Troubleshoot App Errors section of this topic.

Push a SOAP Service

Developers can push Simple Object Access Protocol (SOAP) web services to their PCF deployment by following the procedures in the sections below.

Step 1: Deploy Your Web Service

Perform the following steps to deploy a SOAP web service:

1. Develop the service as an ASMX web service in Microsoft Visual Studio.

2. Publish the service to your local file system.

3. Run `cf push SERVICE-NAME -s windows2012R2 -b hwc_buildpack -u none` to push your service from the directory containing the published web service. Replace `SERVICE-NAME` with the name of your service:

   ```
   $ cf push SERVICE-NAME -s windows2012R2 -b hwc_buildpack -u none
   ```

   The push command must include the `-s` flag to specify the stack as `windows2012R2`, which instructs PCF to run the app in the Windows cell.

   The push command can include the following optional flags:
   - If you are not pushing your service from the directory containing the published web service, use the `-p` flag to specify the path to the directory that contains the published web service.
   - If you do not have a route serving `/`, use the `-u none` flag to disable the health check.

4. Locate the section of the terminal output that displays the URL of the web service:

   ```
   requested state: started
   instances: 1/1
   usage: 1G x 1 instances
   urls: YOUR-WEB-SERVICE.YOUR-DOMAIN
   last uploaded: Thu Nov 17 19:18:19 UTC 2016
   stack: windows2012R2
   buildpack: hwc_buildpack
   ```

Step 2: Modify the WSDL File

Your SOAP web service is now deployed on PCF, but the service's WSDL file contains the incorrect port information. Before an application can consume your web service, either you or the application developer must modify the WSDL file.

Examine the following portion of an example WSDL file:

```
- <wsdl:service name="WebService1">
  - <wsdl:port name="WebService1Soap" binding="tns:WebService1Soap">
      <soap:address location="http://webservice.example.com:62492/WebService1.asmx"/>
    </wsdl:port>
  - <wsdl:port name="WebService1Soap12" binding="tns:WebService1Soap12">
      <soap12:address location="http://webservice.example.com:62492/WebService1.asmx"/>
    </wsdl:port>
- </wsdl:service>
```

The WSDL file provides the port number for the SOAP web service as `62492`. This is the port that the web service listens on in the Garden container, but external applications cannot access the service on this port. Instead, external applications must use port `80`, and the Gorouter routes requests to the web service in the container.

The URL of the web service in the WSDL file must be modified to remove `62492`. With no port number, the URL defaults to port `80`. In the example above, the modified URL would be `http://webservice.example.com/WebService1.asmx`.

SOAP web service developers can resolve this problem in one of two ways:

- Modify the WSDL file by following the instructions in Modify a Web Service's WSDL Using a SoapExtensionReflector from the Microsoft Developers

Network.

- Instruct the developers of external applications that consume the web service to perform the steps in the Consume the SOAP Web Service section of this topic.

Consume the SOAP Web Service

Developers of external applications that consume the SOAP web service can perform the following steps to use a modified version of the WSDL file:

1. In a browser, navigate to the WSDL file of the web service.

   You can reach the WSDL of your web service by constructing the URL as follows: `YOUR-WEB-SERVICE.YOUR-DOMAIN/ASMX-FILE.asmx?wsdl`

   See the following URL as an example: `https://webservice.example.com/WebService1.asmx?wsdl`

2. Download the WSDL file to your local machine.

3. Edit the WSDL file to eliminate the container port, as described in the Modify the WSDL File section of this topic.

4. In Microsoft Visual Studio, right-click on your application in the **Solution Explorer** and select **Add** > **Service Reference**.

5. Under **Address**, enter the local path to the modified WSDL file. For example, `C:\Users\example\wsdl.xml`.

6. Click **OK**. Microsoft Visual Studio generates a client from the WSDL file that you can use in your codebase.

Troubleshoot App Errors

If a .NET app fails to start, consult the following list of errors and their possible solutions:

- `NoCompatibleCell`: Your PCF deployment cannot connect to your Windows cell. See the Troubleshooting Windows Cells topic for information about troubleshooting your Windows cell configuration.
- `Start unsuccessful`: Your app may not contain the required DLL files and dependencies. Ensure that you are pushing from a directory containing your app dependencies, or specify the directory with the `-p` flag. Your app also may be misconfigured. Ensure that your app directory contains either a valid `.exe` binary or a valid `Web.config` file.

Using Apps Manager

The web-based Apps Manager application helps you manage users, organizations, spaces, and applications.

Apps Manager is compatible with current and recent versions of all major browsers. Pivotal recommends using the current version of Chrome, Firefox, Edge, or Safari for the best Apps Manager experience.

Table of Contents

Getting Started with Apps Manager

Page last updated:

Overview

Apps Manager is a web-based tool to help manage organizations, spaces, applications, services, and users. Apps Manager provides a visual interface for performing the following subset of functions available through the Cloud Foundry Command Line Interface (cf CLI):

- **Orgs**: You can create and manage orgs.
- **Spaces**: You can create, manage, and delete spaces.
- **Apps**: You can scale apps, bind apps to services, manage environment variables and routes, view logs and usage information, start and stop apps, and delete apps.
- **Services**: You can bind services to apps, unbind services from apps, choose and edit service plans, and rename and delete service instances.
- **Users**: You can invite new users, manage user roles, and delete users.

To access Apps Manager as the Admin user, see the Logging in to Apps Manager topic.

Understanding Permissions

Your ability to perform actions in Apps Manager depends on your user role and the feature flags that the Admin sets.

The table below shows the relationship between specific org and space management actions and the non-Admin user roles who can perform them. A non-Admin user must be a member of the org and space to perform these actions.

Admin users can perform all of these actions using either the cf CLI or by logging into Apps Manager as an Org Manager, using the UAA Admin credentials.

Space Managers assign and remove users from spaces by setting and unsetting their roles within the space.

Action	CLI command	Org Manager	Space Manager	Org Auditor, Space Developer, or Space Auditor
Create an org	`create-org`	†	†	†
Delete an org	`delete-org`	No	No	No
Rename an org	`rename-org`	Yes	No	No
View org members	`org-users`	Yes	Yes	Yes
Assign user a role in org	`set-org-role`	‡	‡	No
Remove org role from user	`unset-org-role`	‡	‡	No
View space members	`space-users`	Yes	Yes	Yes
Assign user a role in space	`set-space-role`	‡	‡	No
Remove space role from user	`unset-space-role`	‡	‡	No

†Defaults to no. Yes if feature flag `user_org_creation` is set to `true`.

‡Defaults to no. Yes if feature flags `set_roles_by_username` and `unset_roles_by_username` are set to `true`.

Managing Orgs and Spaces Using Apps Manager

Page last updated:

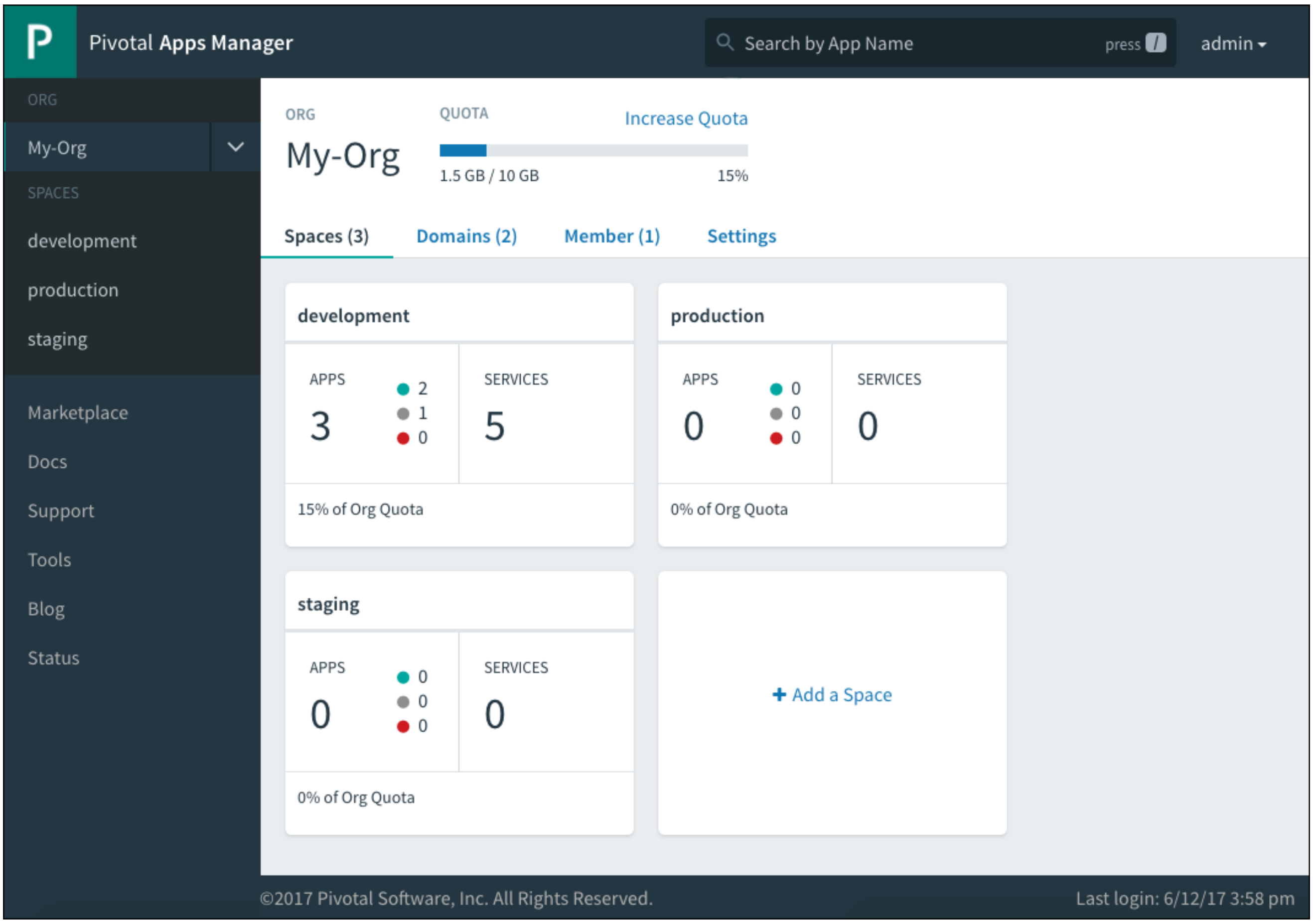
This topic discusses how to view and manage orgs and spaces in Apps Manager.

Note: To manage a space, you must have Space Manager permissions in that space.

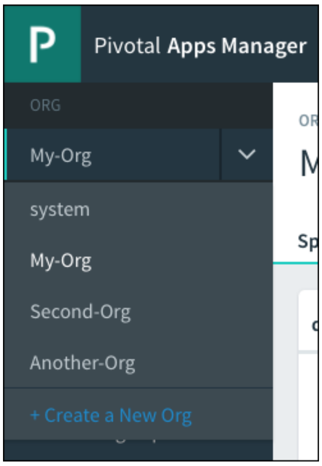
To perform the following steps, you must first log in to Apps Manager with an account that has adequate permissions. See the Understanding Permissions topic for more information.

Manage an Org

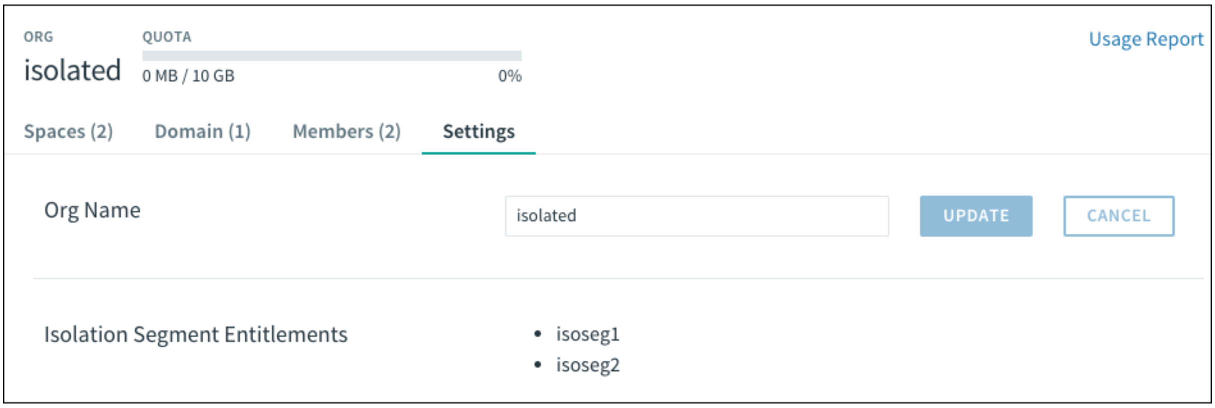
The org page displays the spaces associated with the selected org. The left navigation of Apps Manager shows the current org.



To view spaces in a different org, use the drop-down menu to change the org.



To view the page for a particular space, click the space on the org page or on the left navigation. To create a new space, click **Add a Space** at the bottom of the org page.

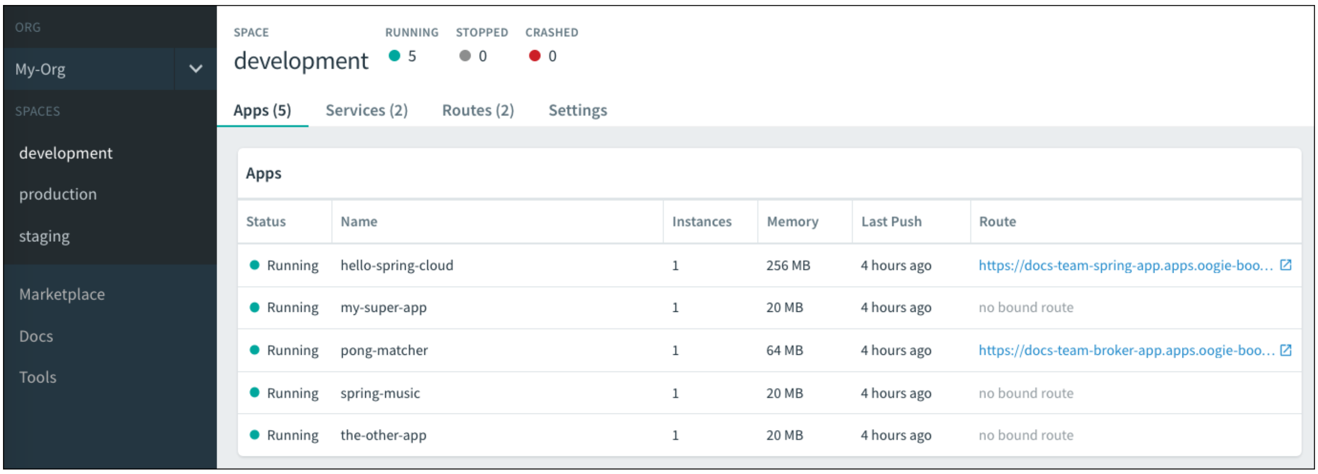


To rename the current org, click the **Settings** tab. Enter the new org name in the **Org Name** section and click **Update**.

From the **Settings** tab, you can also view the spaces assigned to your isolation segment in the **Isolation Segment Entitlements** section. For more information, see Isolation Segments.

Manage a Space

The space page displays the apps, service instances, and routes associated with the selected space.



Apps

The **Apps** tab shows the **Status**, the **Name**, the number of **Instances**, the amount of **Memory** available, the time since the **Last Push**, and the **Route** for each app.

Apps

Status	Name	Instances	Memory	Last Push	Route
Running	hello-spring-cloud	1	256 MB	4 hours ago	https://docs-team-spring-app.apps.oo…
Running	my-super-app	1	20 MB	4 hours ago	no bound route

Services

The **Services** list shows the **Service**, the **Name**, the number of **Bound Apps**, and the **Plan** for each service instance. If you want to add a service to your space, click **Add Service**. For more information about configuring services, see the Services Overview topic.

Services ADD A SERVICE

Service	Name	Bound Apps	Plan
App Autoscaler	my-autoscaler	0	free - Standard

Routes

From the **Routes** tab, you can view and delete routes associated with your space. For each **Route**, the tab shows **Route Service** and **Bound Apps**. Refer to the Route Services topic to learn more about route services.

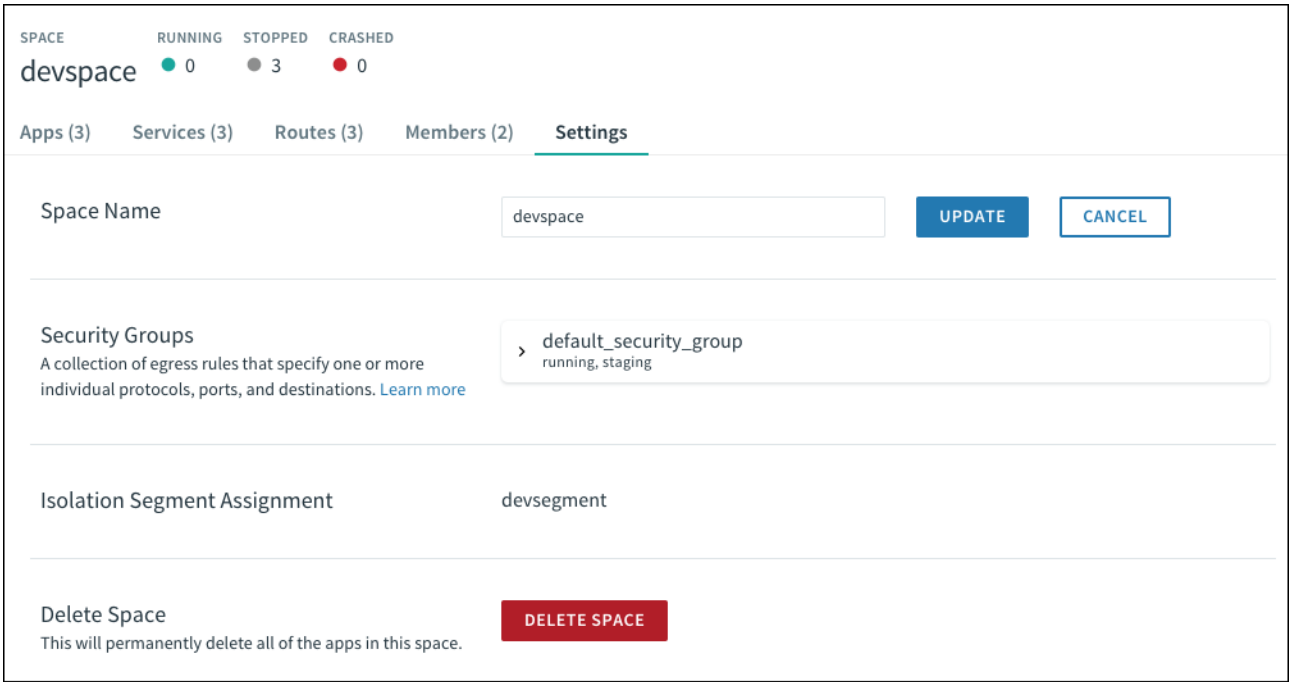
Routes

Route	Route Service	Bound Apps
docs-team-broker-app.apps.oogie-boogie…	my-service	pong-matcher
docs-team-spring-app.apps.oogie-boogie.…	no bound service	hello-spring-cloud

Settings

From the **Settings** tab, you can do the following:

- Modify the space name by entering a new name and clicking **Update**.
- View the Application Security Groups (ASGs) associated with the space in the **Security Groups** section.
- View the space's isolation segment assignment in the **Isolation Segment Assignment** section. For more information, see Isolation Segments.
- Delete the space by clicking **Delete Space**.


SPACE
devspace
RUNNING
0
STOPPED
3
CRASHED
0
Apps (3)
Services (3)
Routes (3)
Members (2)
Settings
Space Name
devspace
UPDATE
CANCEL
Security Groups
A collection of egress rules that specify one or more individual protocols, ports, and destinations. Learn more
default_security_group
running, staging
Isolation Segment Assignment
devsegment
Delete Space
This will permanently delete all of the apps in this space.
DELETE SPACE

Managing User Roles with Apps Manager

Page last updated:

> **Note**: The procedures described here are not compatible with using SAML or LDAP for user identity management. To create and manage user accounts in a SAML- or LDAP-enabled Cloud Foundry deployment, see Adding Existing SAML or LDAP Users to a Pivotal Cloud Foundry Deployment.

Cloud Foundry uses role-based access control, with each role granting the permissions in either an org or an application space.

A user account can be assigned one or more roles.

The combination of these roles defines the actions a user can perform in an org and within specific app spaces in that org.

To view the actions that each role allows, see the Organizations, Spaces, Roles, and Permissions topic. For example, to assign roles to user accounts in a space, you must have Space Manager role assigned to the user in that space.

You can also modify permissions for existing users by adding or removing the roles associated with the user account. User roles are assigned on a per-space basis, so you must modify the user account for each space that you want to change.

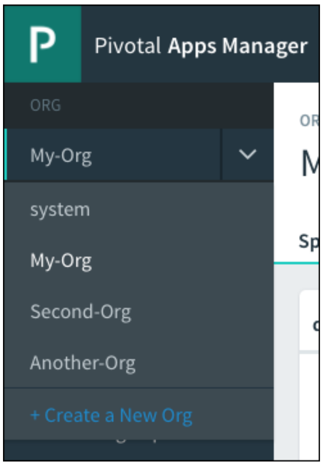
Admins, Org Managers, and Space Managers can assign user roles with Apps Manager or with the Cloud Foundry Command Line Interface (cf CLI). For more information, see the Users and Roles section of the *Getting Started with the cf CLI* topic.

Manage Org Roles

Valid org roles are Organization Manager and Organization Auditor.

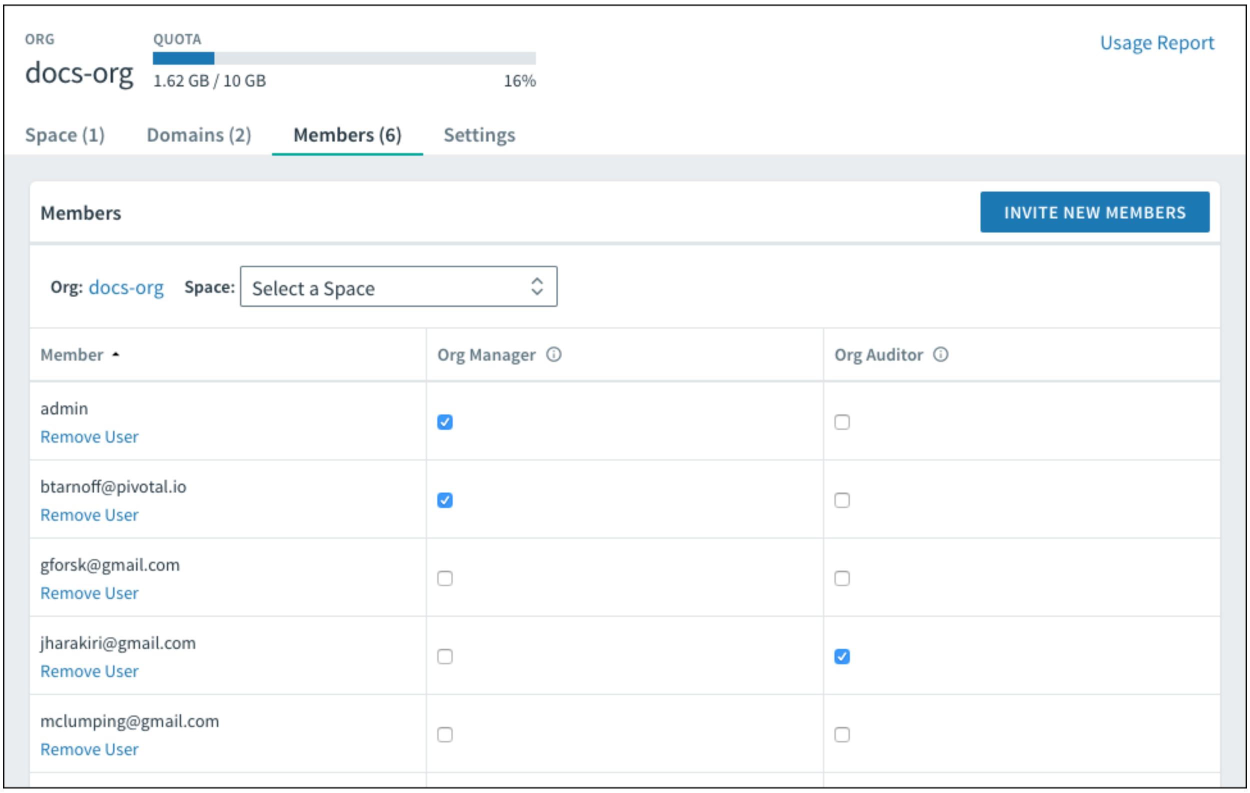
To grant or revoke org roles, follow the steps below.

1. In the Apps Manager navigation on the left, the current org is highlighted. Click the drop-down menu to view other orgs belonging to the account.



2. Use the Apps Manager navigation to select an org.

3. Click the **Members** tab. Edit the roles assigned to each user by selecting or clearing the checkboxes under each user role. Apps Manager saves your changes automatically.



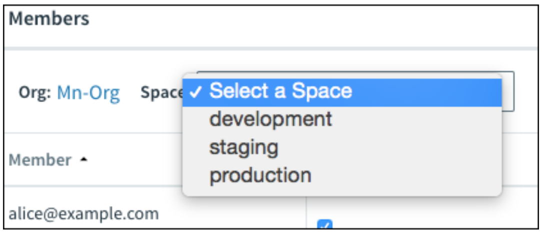
4. The **Members** panel displays all members of the org. Select a checkbox to grant an org role to a user, or clear a checkbox to revoke a role from a user.

Manage App Space Roles

Valid app space roles are Space Manager, Space Developer, and Space Auditor.

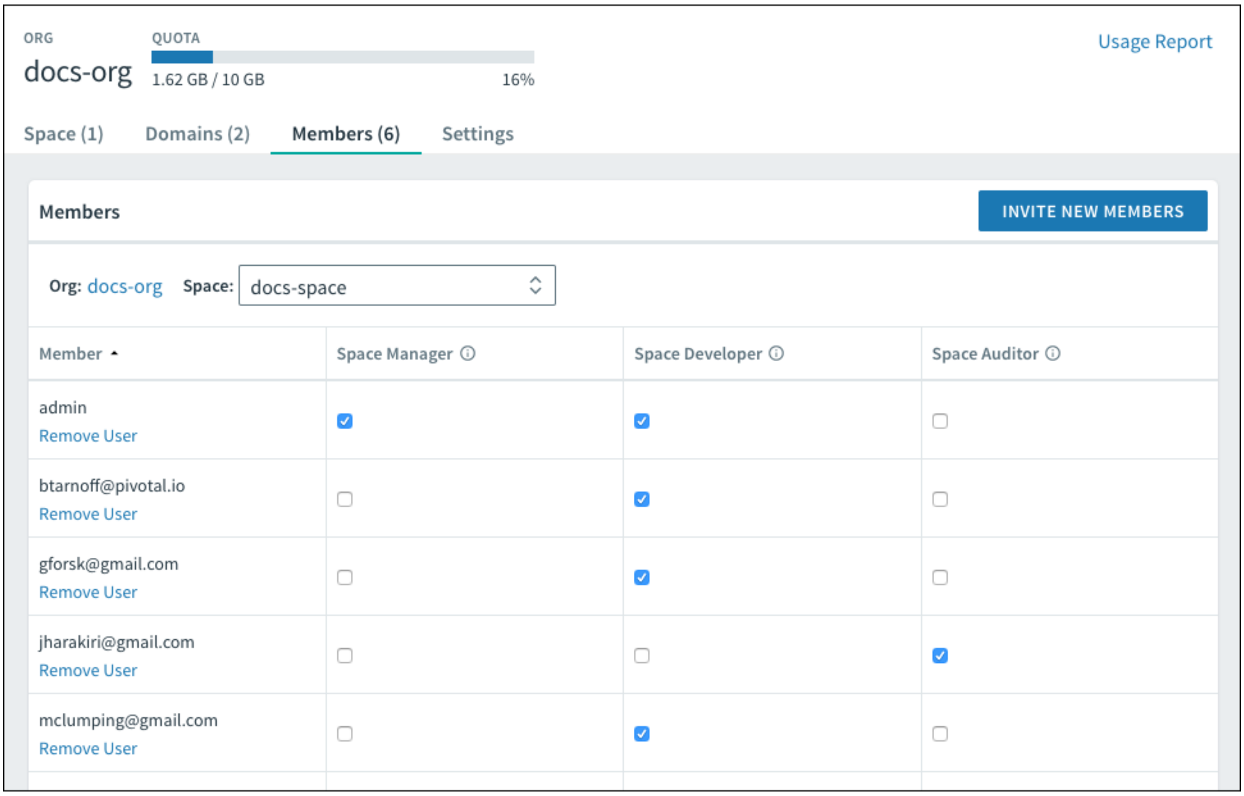
To grant or revoke app space roles, follow the steps below.

1. In the **Members** tab of an org, click the drop-down menu to view spaces in the org.



2. Use the drop-down menu to select a space.

3. The **Members** panel displays all members of the space. Select a checkbox to grant an app space role to a user, or clear a checkbox to revoke a role from a user.

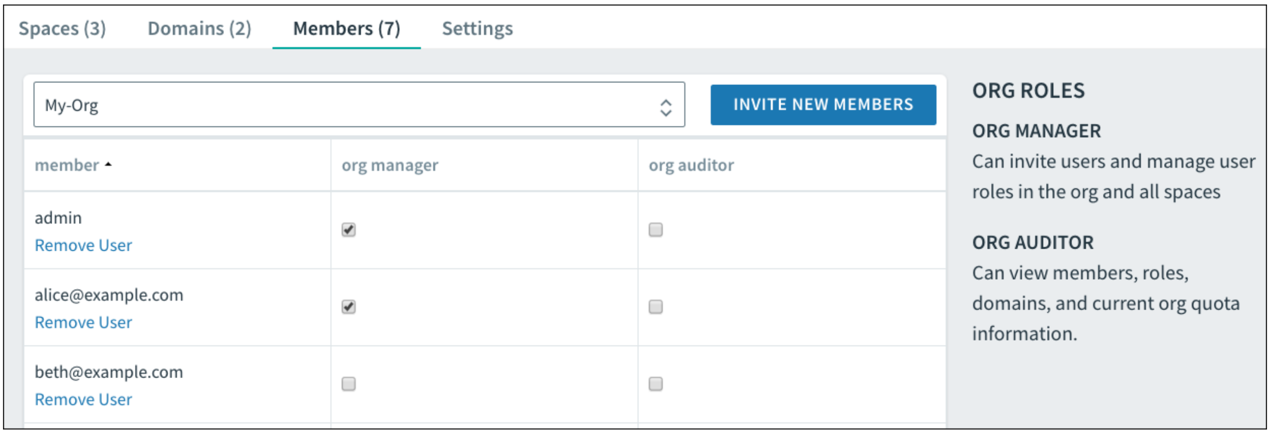


- **Space Managers** can invite and manage users and enable features for a given space. Assign this role to managers or other users who need to administer the account.
- **Space Developers** can create, delete, and manage applications and services, and have full access to all usage reports and logs. Space Developers can also edit applications, including the number of instances and memory footprint. Assign this role to app developers or other users who need to interact with applications and services.
- **Space Auditors** have view-only access to all space information, settings, reports, and logs. Assign this role to users who need to view but not edit the application space.

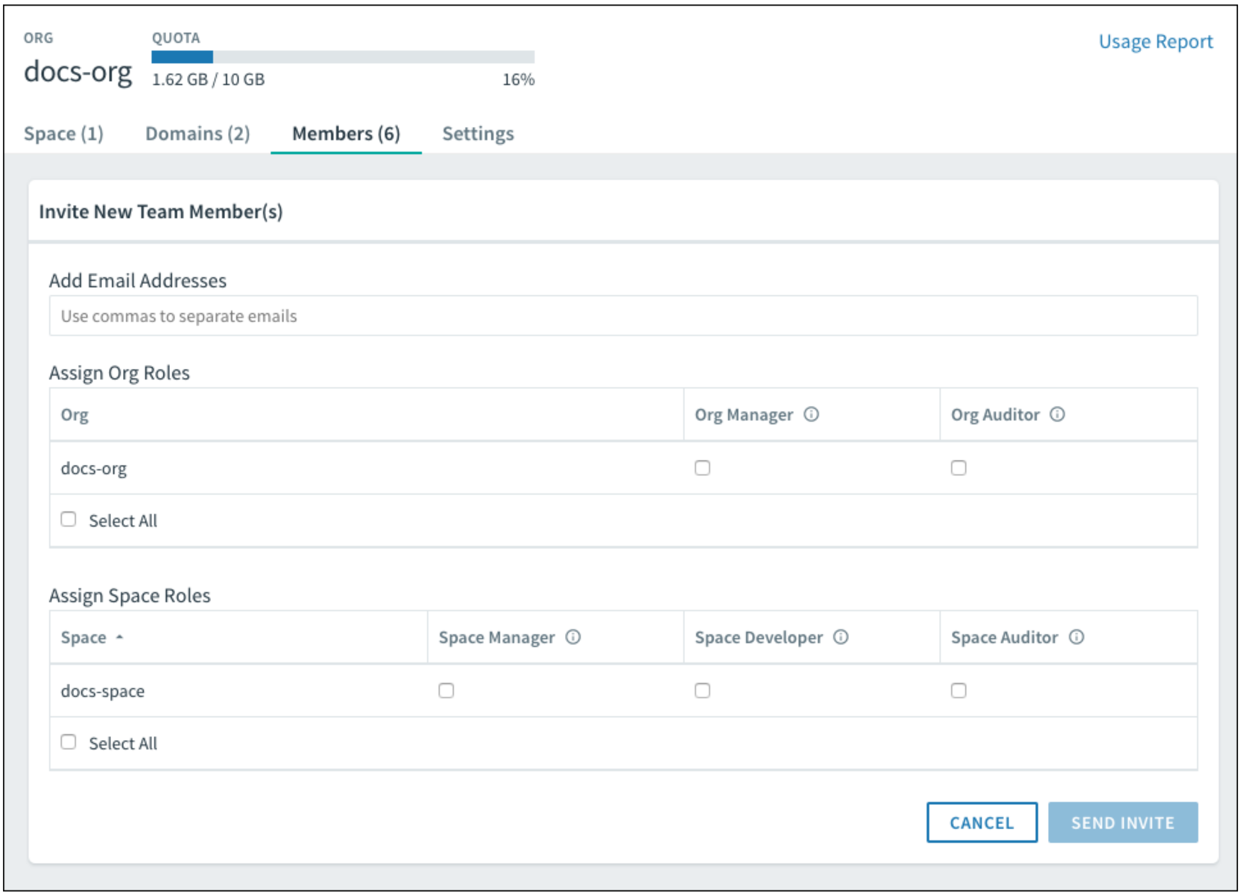
Invite New Users

Note: The **Enable Invitations** checkbox in the Apps Manager section of the PAS tile must be selected to invite new users.

1. On the Org dashboard, click the **Members** tab.



2. Click **Invite New Members**. The **Invite New Team Member(s)** form appears.



3. In the **Add Email Addresses** text field, enter the email addresses of the users that you want to invite. Enter multiple email addresses as a comma-delimited list.

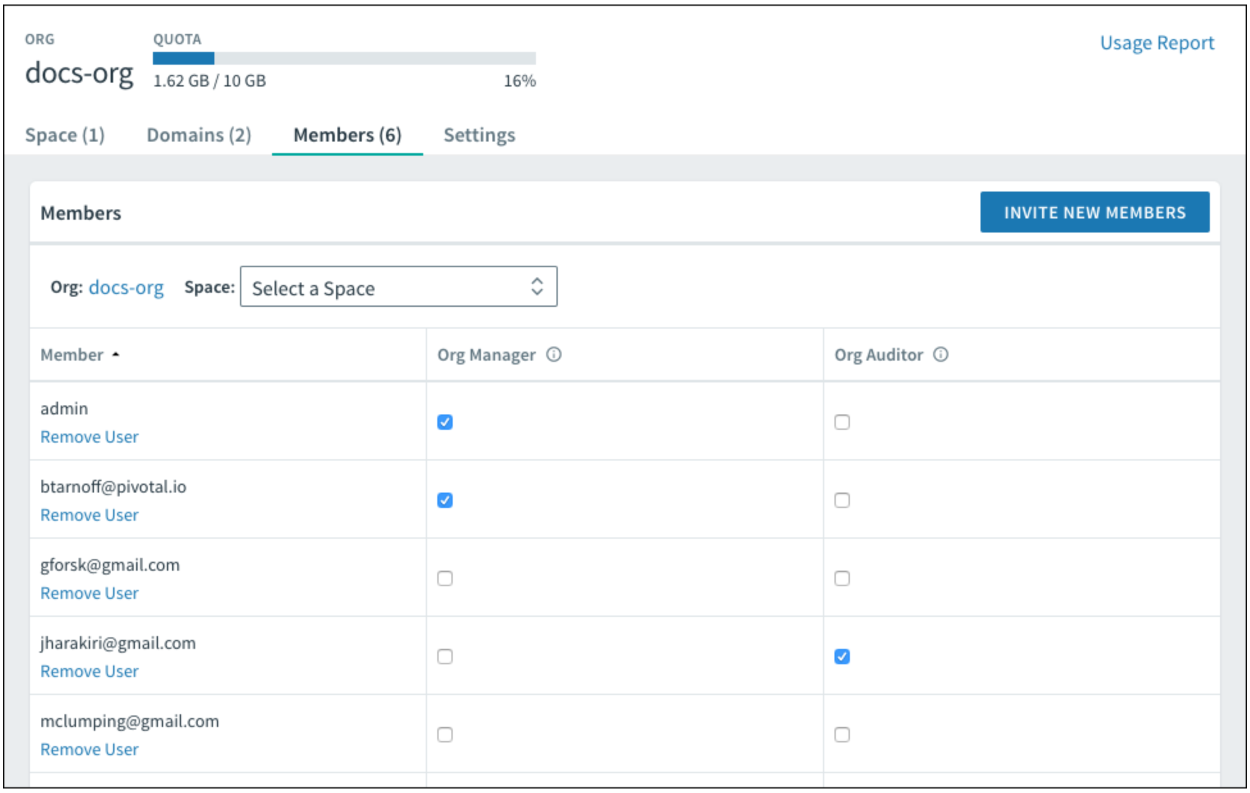
4. The **Assign Org Roles** and **Assign Space Roles** tables list the current org and available spaces with checkboxes corresponding to each possible user role. Select the checkboxes that correspond to the permissions that you want to grant to the invited users.

5. Click **Send Invite**. The Apps Manager sends an email containing an invitation link to each email address that you specified.

Remove a User From an Org

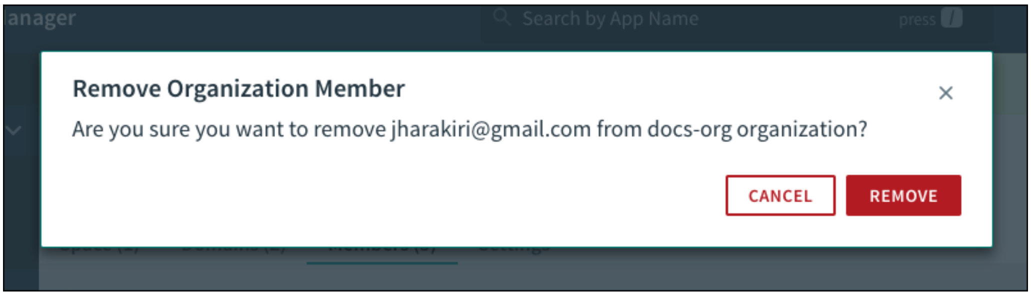
Removing a user from org also removes them from all spaces in the org.

1. On the Org dashboard, click the **Members** tab.



2. Locate the user account that you want to remove.

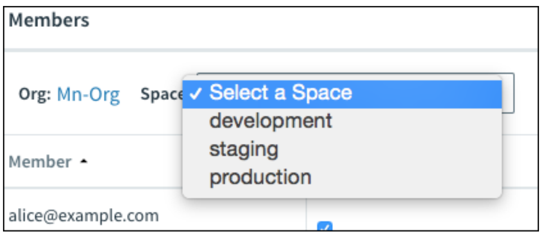
3. Under the user's email address, click on the **Remove User** link. A warning dialog appears.



4. Click **Remove** to confirm user account deletion from the org.

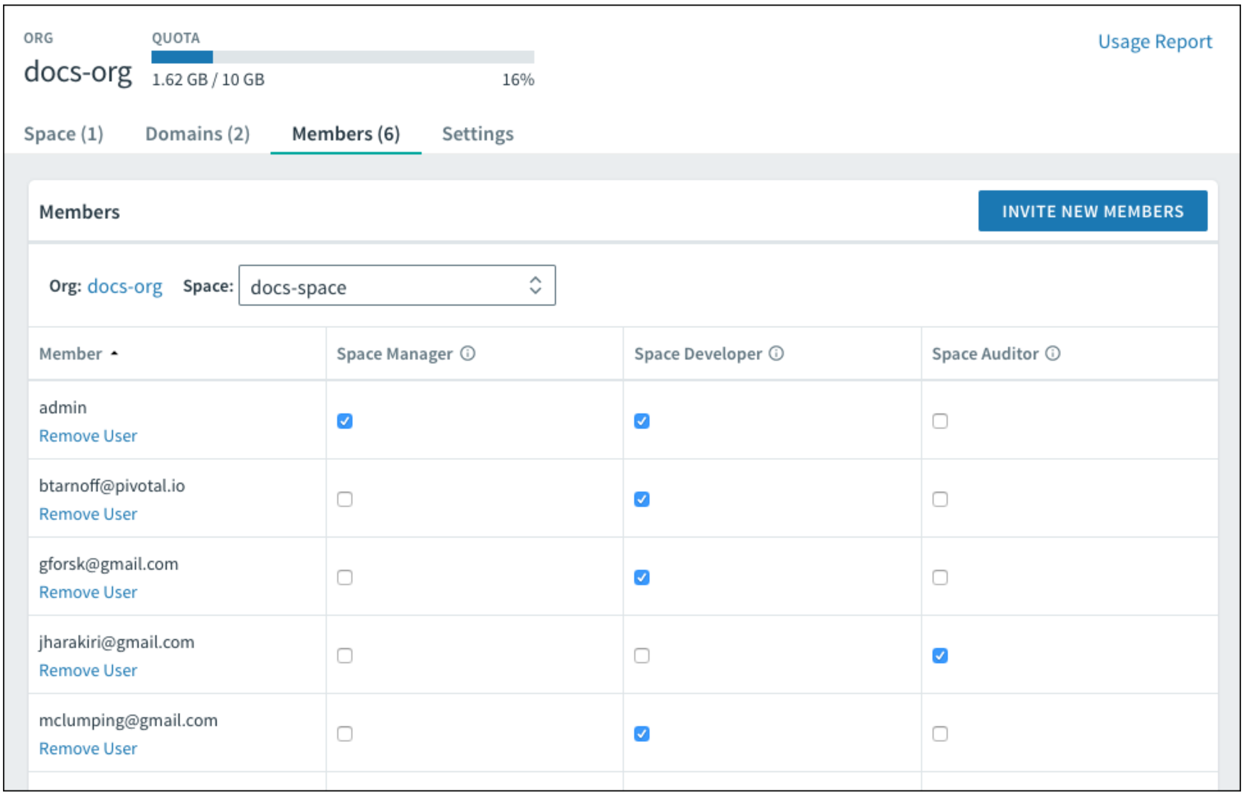
Remove a User From a Space

1. In the **Members** tab of an org, click the drop-down menu to view spaces in the org.

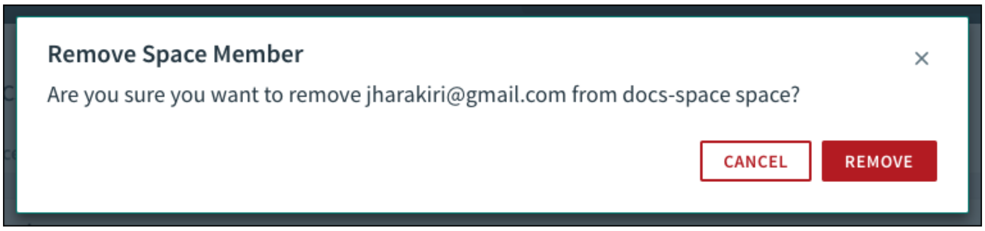


2. Select the space you are removing members from.

3. The **Members** panel displays all members of the space. Locate the user account that you want to remove.



4. Under the user's email address, click on the **Remove User** link. A warning dialog appears.



5. Click **Remove** to confirm user account deletion from the space.

Managing Apps and Service Instances Using Apps Manager

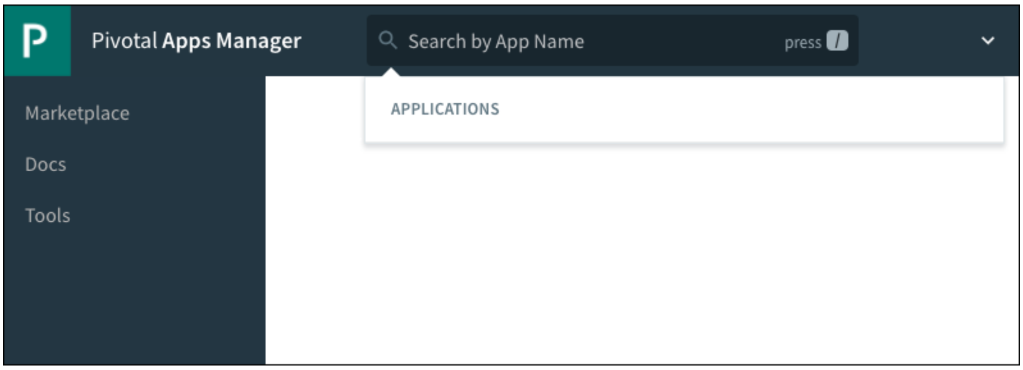
Page last updated:

This topic discusses how to view and manage apps and service instances associated with a space using Apps Manager.

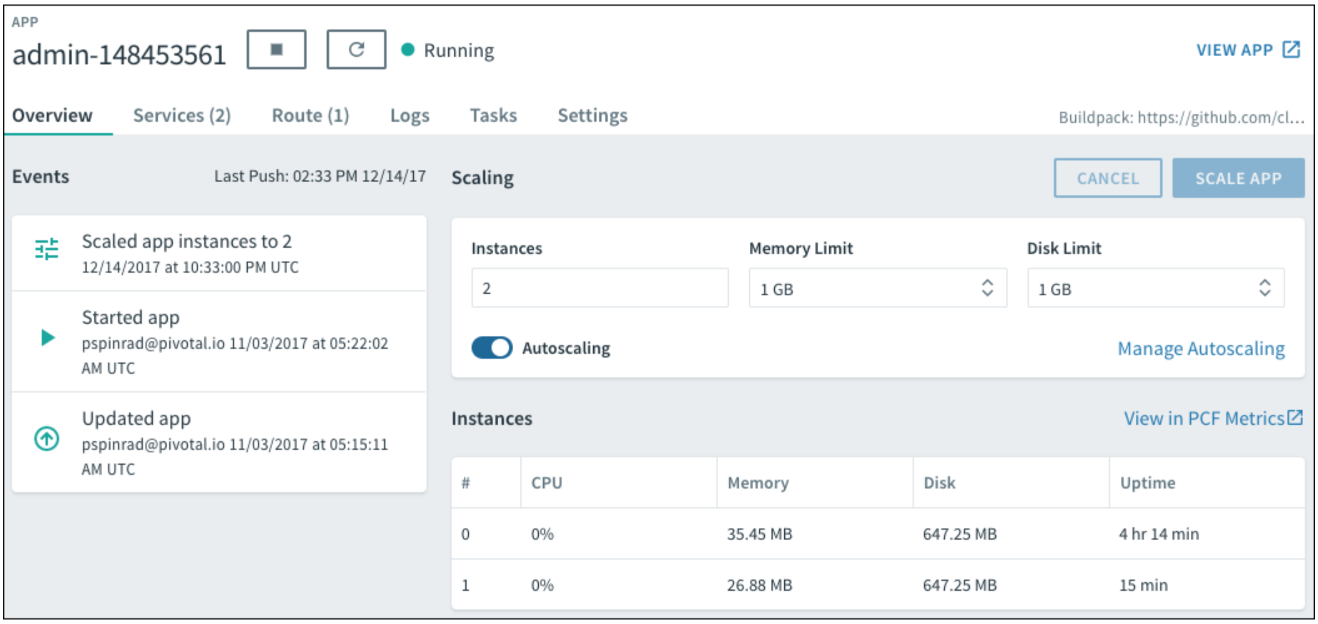
To perform the following steps, you must first log in to Apps Manager with an account that has adequate permissions. See the Understanding Permissions topic for more information.

Manage an App

On the space page, click the app you want to manage. You can search for the app by entering its name in the search bar.



From the app page, you can scale apps, bind apps to services, manage environment variables and routes, view logs and usage information, start and stop apps, and delete apps.



Start or Stop an App

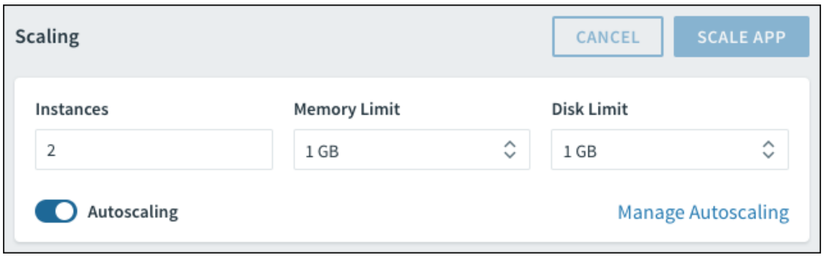
1. To stop an app, click the stop button next to the name of the app. Click **Stop** in the pop-up to confirm.

2. To restart a stopped app, click the play button next to the name of the app.

3. To restart a running app, click the restart button next to the name of the app. Click **Restart** in the pop-up to confirm.

Scale an App

From the app **Overview** pane, Space Developers can scale an app manually or configure App Autoscaler to scale it automatically.



Scale an App Manually

1. Under **Scaling**, adjust the number of **Instances**, the **Memory Limit**, and the **Disk Limit** as desired.

2. Click **Scale App**.

Configure App Autoscaler

1. Use the **Autoscaling** slider to enable App Autoscaler.

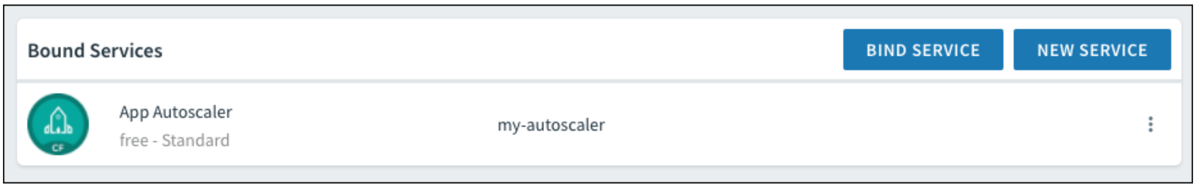
2. Click **Manage Autoscaling** to open App Autoscaler.

3. See the Configure Autoscaling for an App section of the *Scaling an Application Using Autoscaler* topic for how to configure your App Autoscaler to scale automatically based on rules or a schedule.

Bind or Unbind a Service

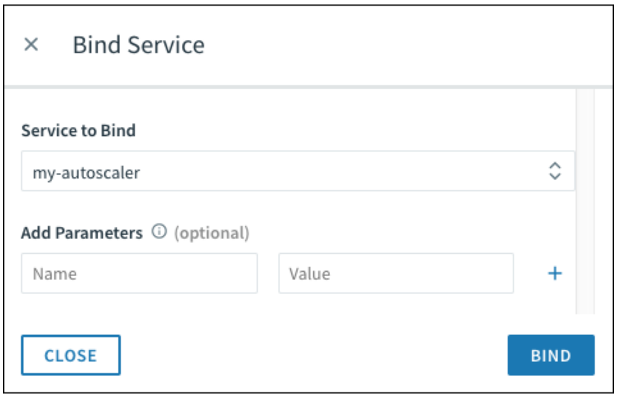
1. Click **Services**.

2. To bind your app to a service, click **Bind Service**.



3. To bind your app to an an existing service instance, perform the following steps:

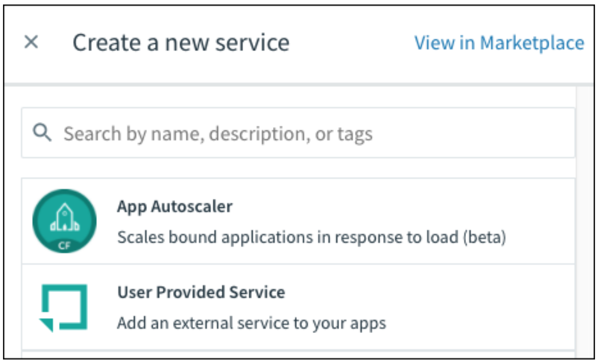
    a. Click **Bind Service**.



    b. Under **Service to Bind**, select the service instance from the dropdown menu.
    c. Optionally add any parameters under **Add Parameters**.
    d. Click **Bind**.

4. To bind your app to a new service instance, perform the following steps:

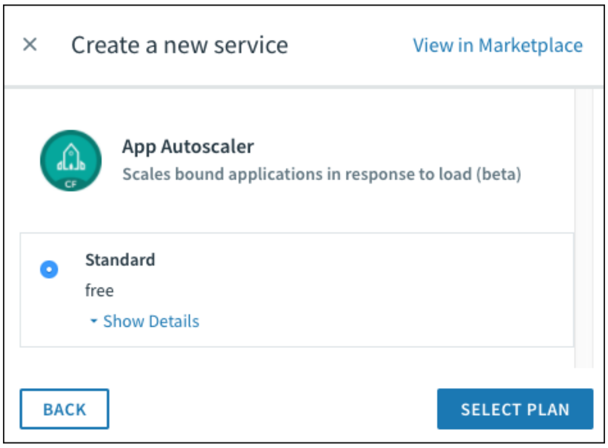
   a. Click **New Service**.



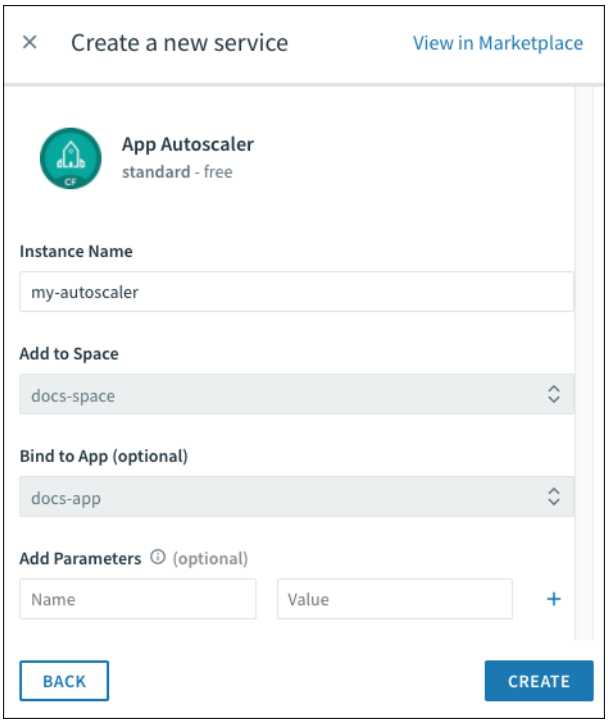
.

> **Note**: If you prefer to create the new service instance in the Marketplace, you can click **View in Marketplace** at any time.

   b. Click the service.



   c. Select a plan and click **Select Plan**.



   d. Under **Instance Name**, enter a name for the instance.

   e. Optionally add any parameters under **Add Parameters**. The service may provide different parameter options, or a JSON editor for you to define the parameters. For more information, consult the documentation for the service.

   f. Click **Create**.

5. To unbind your app from a service instance, locate the service instance in the **Bound Services** list and click the three-dot icon on the far right. Select **Unbind** from the dropdown menu.

Map or Unmap Routes

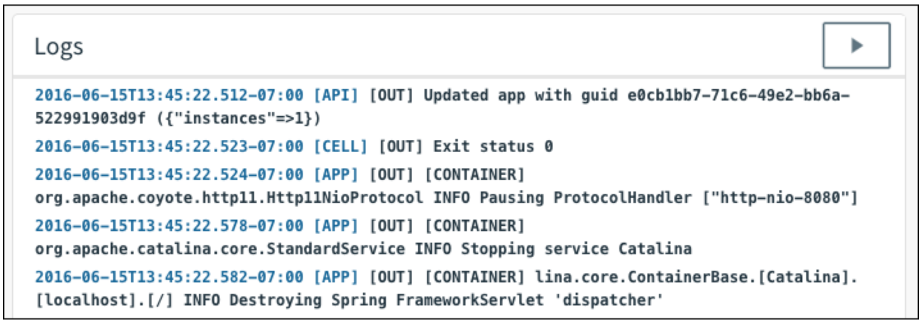
1. Click **Routes**.

2. The page displays the routes associated with the app. To add a new route, click **Map a Route**.

3. Enter the route and click **Map**.

4. To unmap a route, locate the route from the list and click the red `x`. Click **Unmap** in the pop-up to confirm.

View Logs



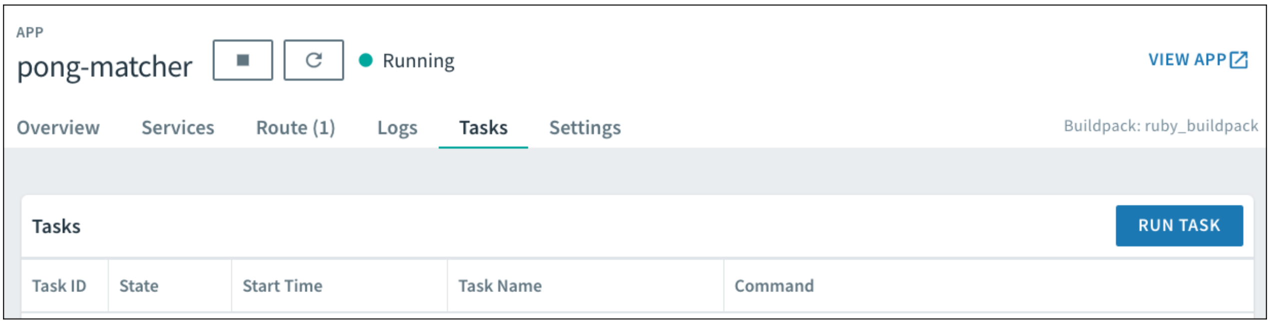
1. Click **Logs** to view the logs for the app.

2. Click the play button to view a live version of the logs.

View Tasks

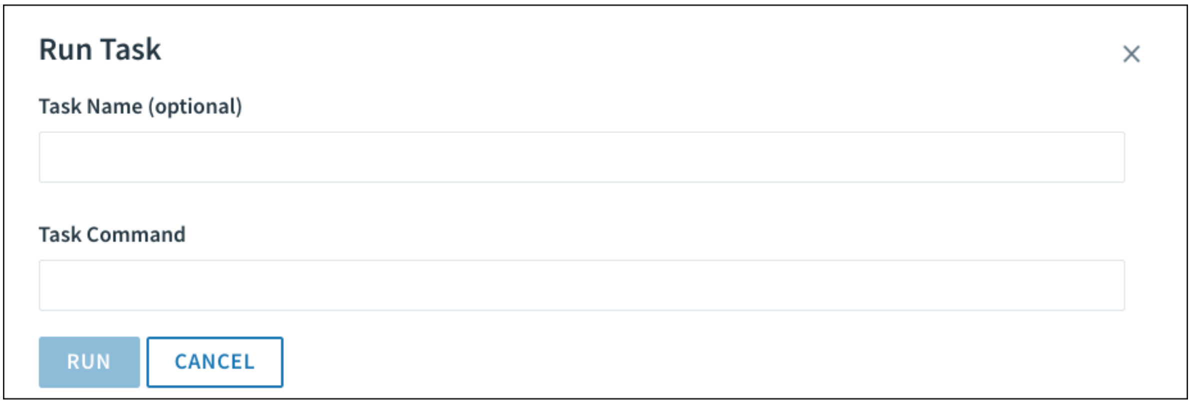
1. Click the **Tasks** tab within Apps Manager.

2. This page displays a table containing **Task ID**, **State**, **Start Time**, **Task Name**, and **Command**.



Run a Task

1. Click **Run Task** to create a task.



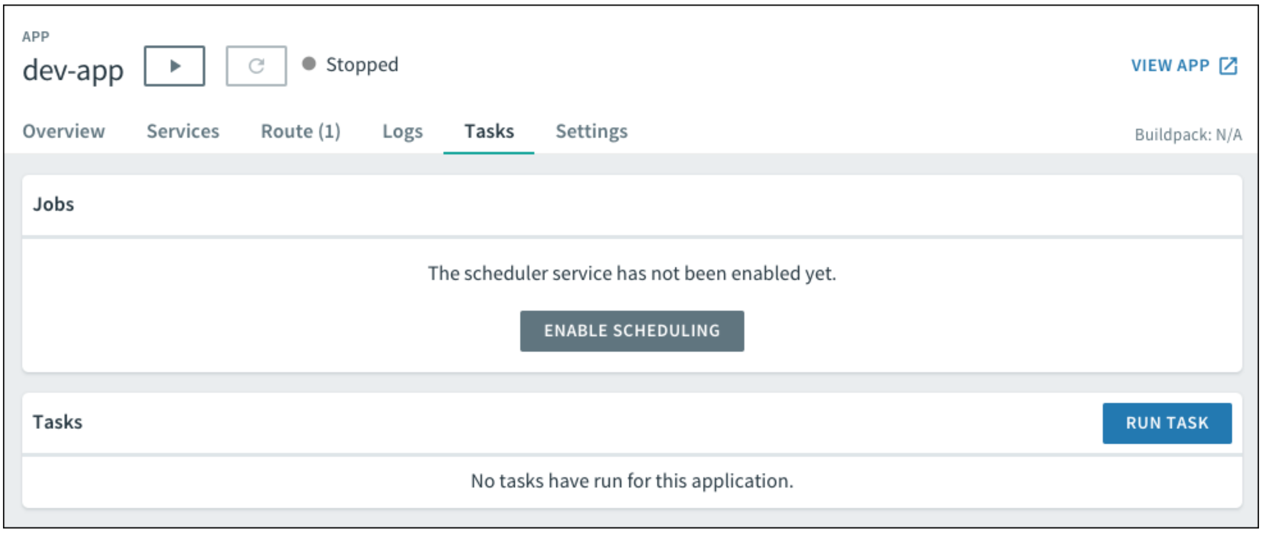
2. (Optional) Enter a **Task Name**.

3. Enter the **Task Command**.

4. Click **Run**.

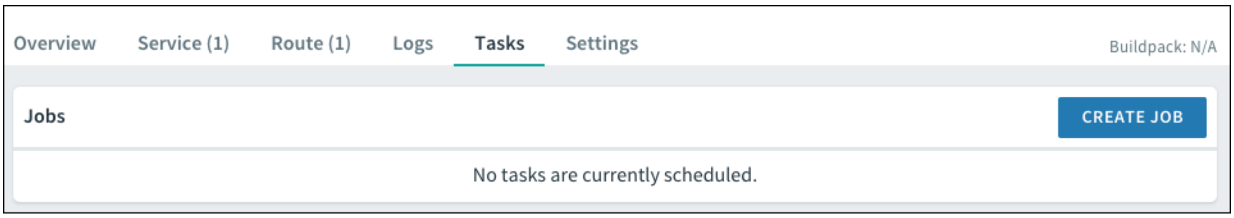
Enable Task Scheduling

In the **Tasks** tab, click **Enable Scheduling** to bind the PCF Scheduler service to your app. For more about the PCF Scheduler, see Scheduling Jobs.



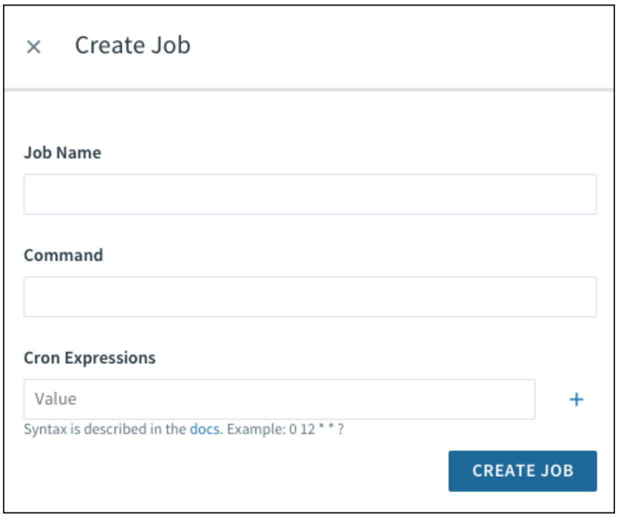
Schedule a Task

1. Navigate to the **Tasks** tab.



2. Click **Create Job** to schedule a task.

3. Enter a **Job Name**.



4. Enter a **Command**.

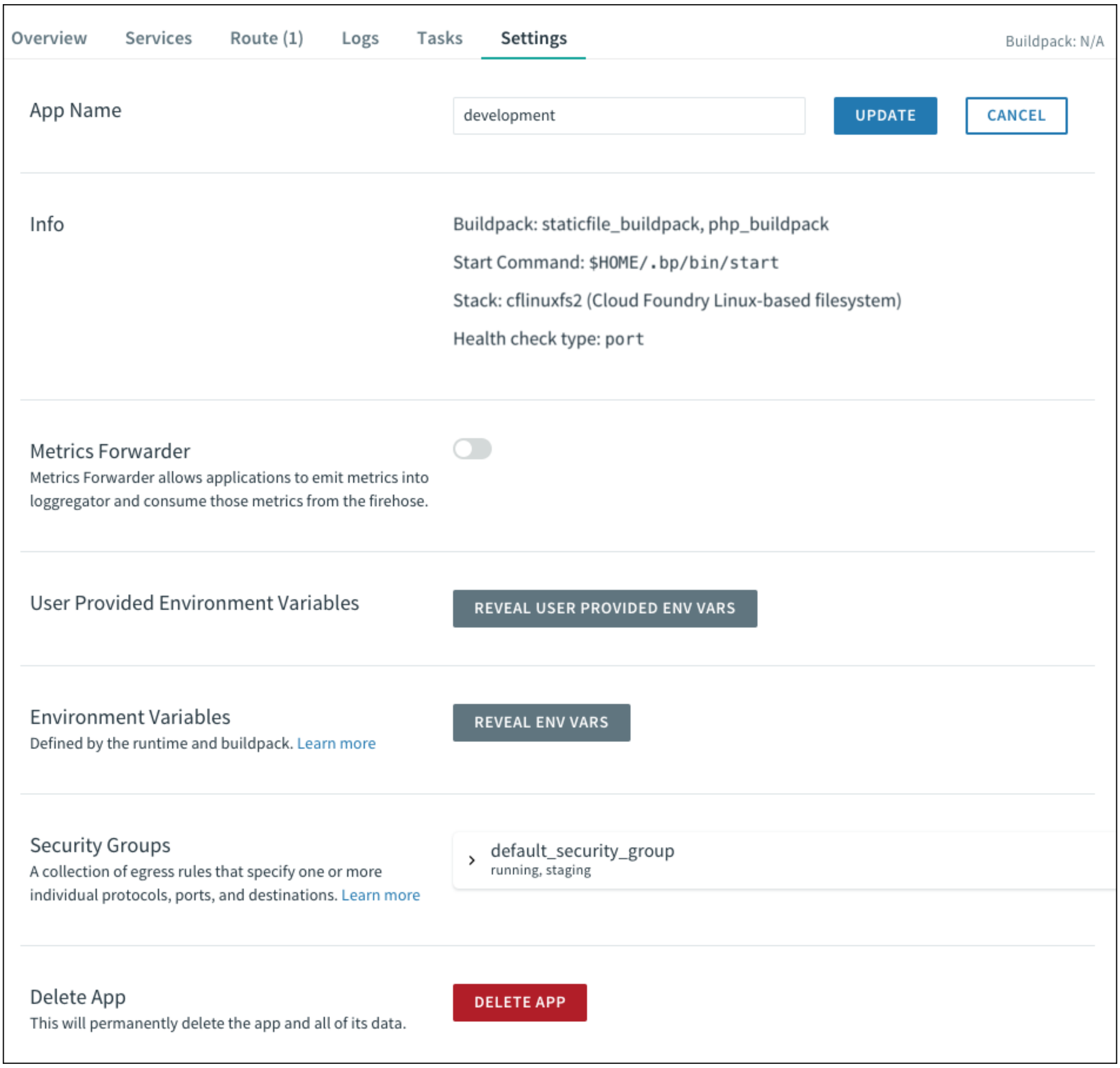
5. Enter one or more **Cron Expressions** for your desired task schedule or schedules. See Schedule a Job for more information on cron expression syntax.

6. Click **Create Job**

View Settings

Click the **Settings** tab. In this tab you can do the following:

- Rename the app.
- View information about the buildpack(s), start command, stack, and health check.
- Enable the **Metrics Forwarder** service. The service allows the app to emit metrics into Loggregator. For more information about metrics forwarding, see Emitting Metrics to Metrics Forwarder for PCF.
- View or add Environment Variables associated with the app.
- View the Application Security Groups (ASGs) associated with the app.
- Delete the app.



View Health Checks

Follow the steps below to view information about a health check you have configured for your app. For more information, see the Using Application Health Checks topic.

1. Click the **Settings** tab.

2. Under **Info**, find the **Health check type**. If your health check type is HTTP, Apps Manager also displays the **Health check endpoint**.

   Health check type: http

   Health check endpoint: /

View or Add Environment Variables

Follow the steps below to add a user-provided environment variable.

1. Click the **Settings** tab.

2. Click **Reveal User Provided Env Vars**.

3. Enter the **Name** and **Value** of the variable.

4. Click **Save**.

To view all environment variables, click **Reveal Env Vars**.

Note: Changes to environment variables, service bindings, and service unbindings require restarting the app to take effect. You can restart the app from the Apps Manager or with the Cloud Foundry Command Line Interface `cf restage` command.

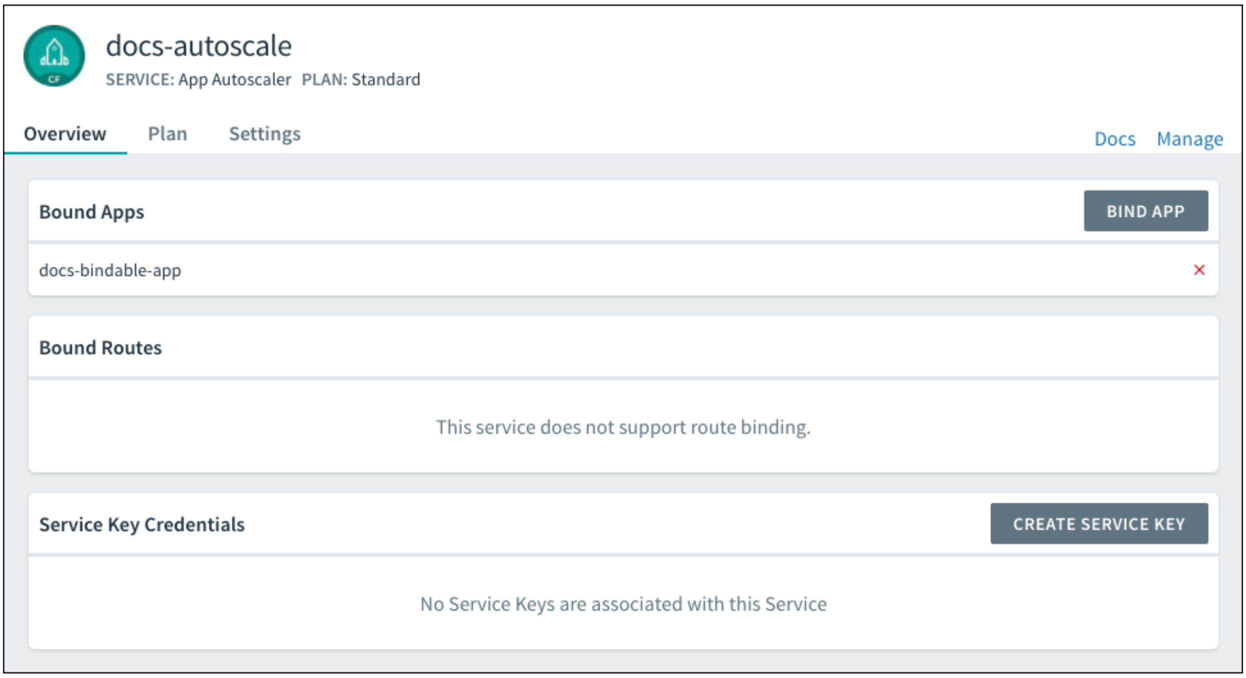
Manage a Service Instance

From the **Services** tab on the space page, you can bind or unbind apps, bind or unbind routes, view or change your service plan, manage service keys, and rename or delete your service instance.

For services that use on-demand brokers, the service broker will create, update, or delete the service instance in the background and notify you when it finishes.

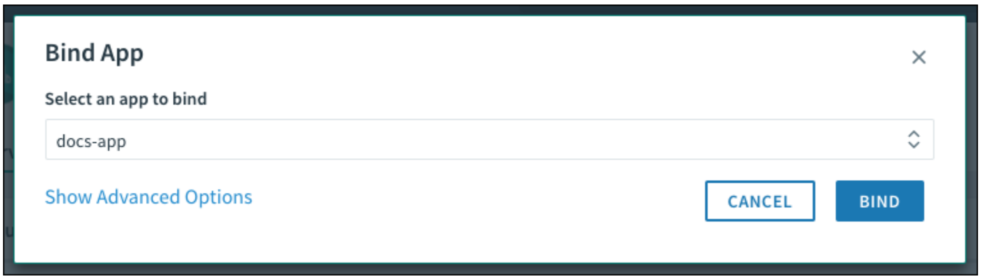
Bind an App

1. From the space page **Services** tab, click the service instance you want to bind to an app.



2. Click **Bind App**. A popup appears.

3. In the **Bind App** popup, select the app you want to bind to your service instance.

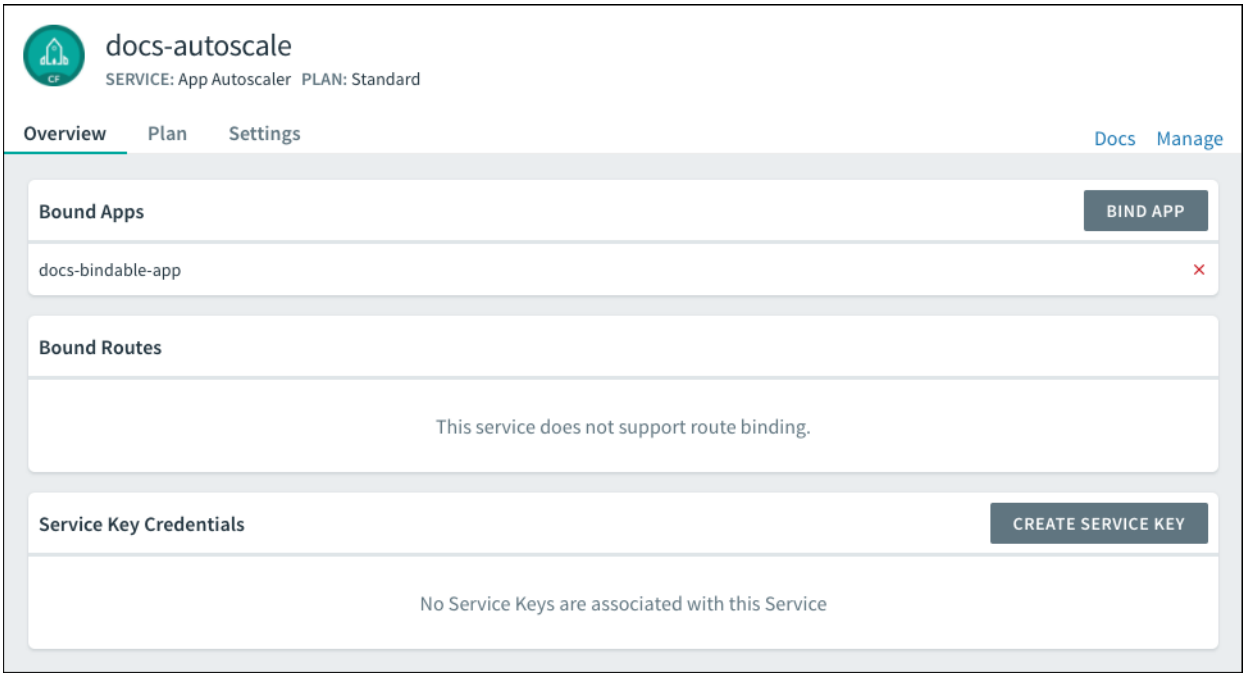


4. (Optional) To attach parameters to the binding, click **Show Advanced Options**. Under **Arbitrary Parameters**, enter any additional service-specific configuration in the **Name** and **Value** fields.

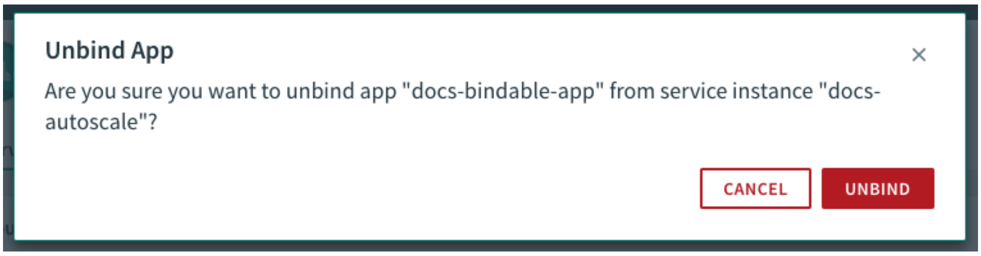
5. Click **Bind**.

Unbind an App

1. From the space page **Services** tab, click the service instance you want to unbind from an app.



2. Locate the app under **Bound Apps** and click the red × on the right. An **Unbind App** popup appears.

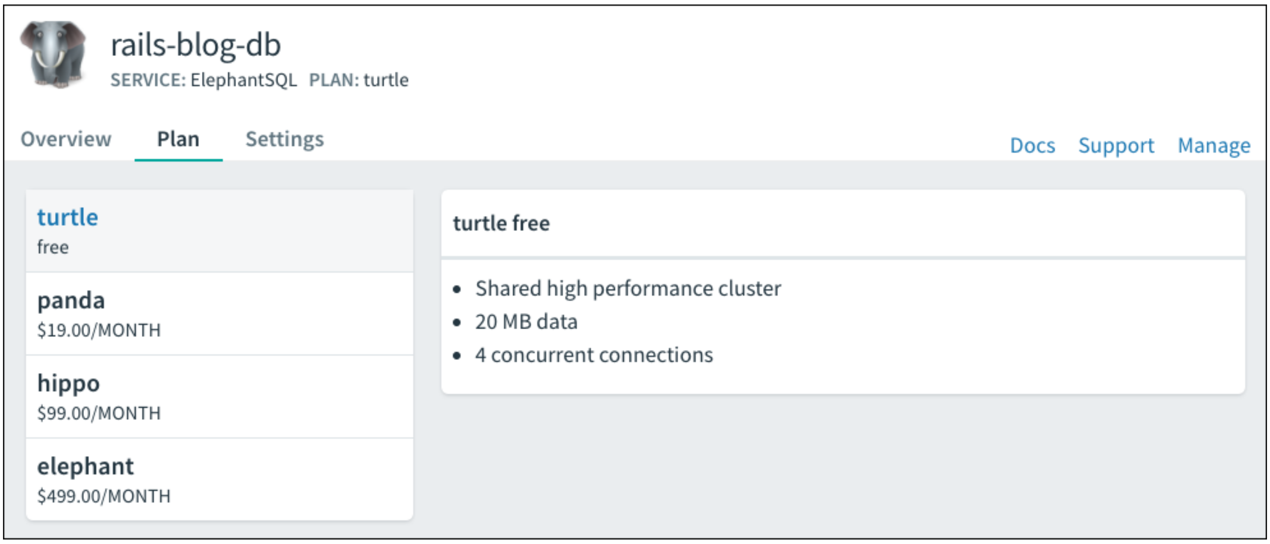


3. Click **Unbind** to confirm.

View or Change Your Service Plan

1. From the space page **Services** tab, click the service instance you want to view or change the plan for.

2. Click **Plan**.



3. Review your current plan information.

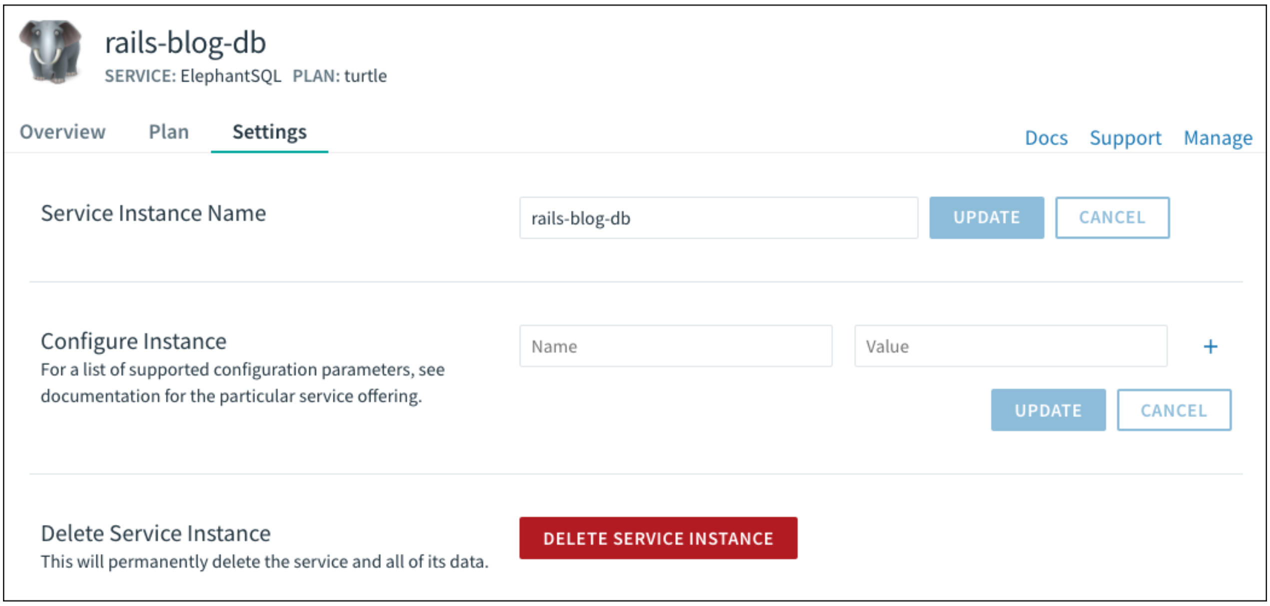
4. To change your plan, select a new plan from the list and click **Select This Plan** or **Upgrade Your Account**.

Note: Not all services support upgrading. If your service does not support upgrading, the service plan page only displays the selected plan.

Rename or Delete Your Service Instance

1. From the space page **Services** tab, click the service instance you want to rename or delete.

2. Click **Settings**.



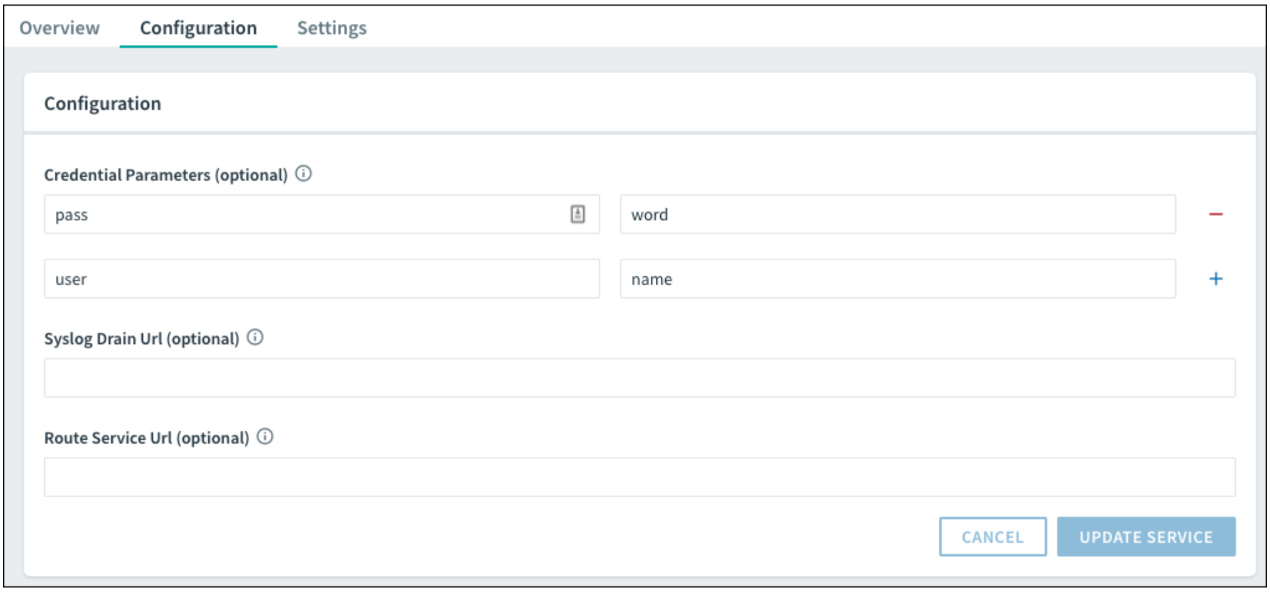
- To change the service instance name, enter the new name and click **Update**.
- To add configuration parameters to the service instance, enter the parameters in the **Name** and **Value** fields, then click **Update**.
- To delete the service instance, click **Delete Service Instance**.

Note: The service broker supports creating, updating, and deleting service instances asynchronously. When the service broker completes one of these operations, a status banner appears in Apps Manager.

Update Your User-Provided Service Instance

Follow the steps below to update an existing user-provided service instance. You can create a user-provided service instance from the Marketplace. For more information, see the User-Provided Service Instances topic.

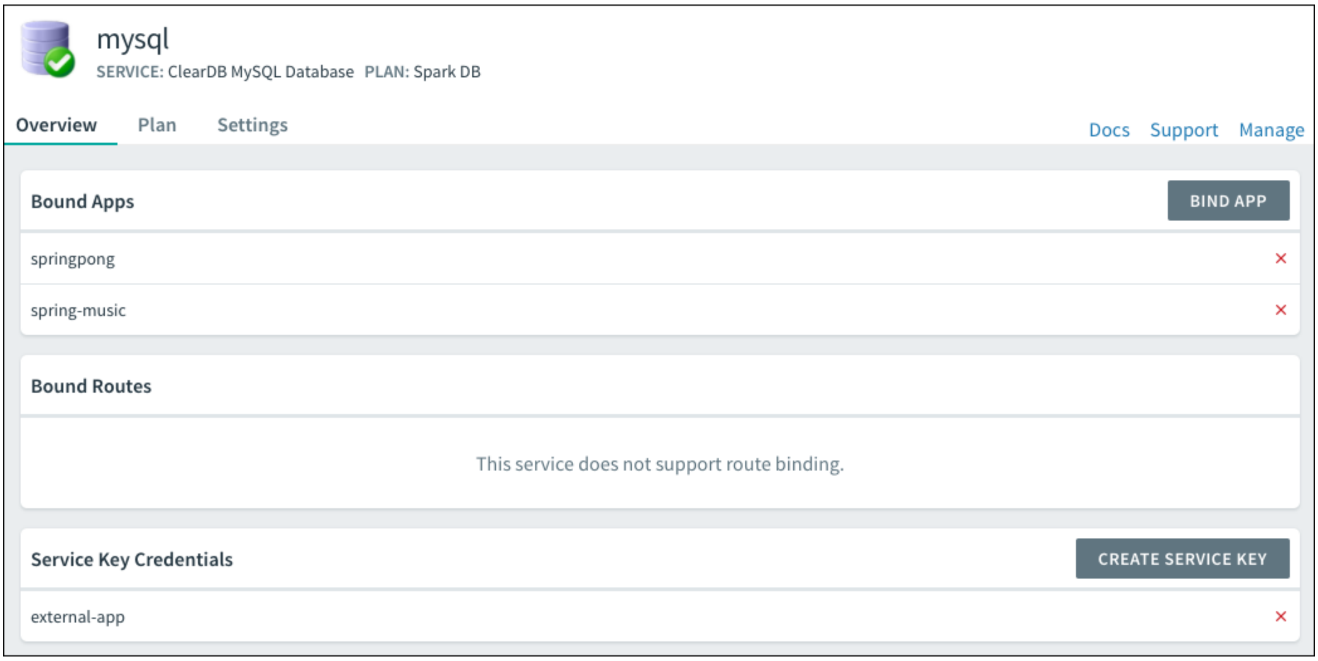
1. Click **Configuration**. This tab only appears for user-provided service instances.



2. Enter your **Credential Parameters**, **Syslog Drain Url**, and **Route Service Url**, and click **Update Service**.

Manage Service Keys

On the space page, click **Services**, then click the service instance that you want to manage service keys for. This directs you to the service instance **Overview** page, where you can generate a new service key, get the credentials for a service key, and delete a service key.



Generate a Service Key

Follow the steps below to generate a service key.

1. In the **Service Key Credentials** section, click **Create Service Key**.

2. Edit the **Service Key Name**.

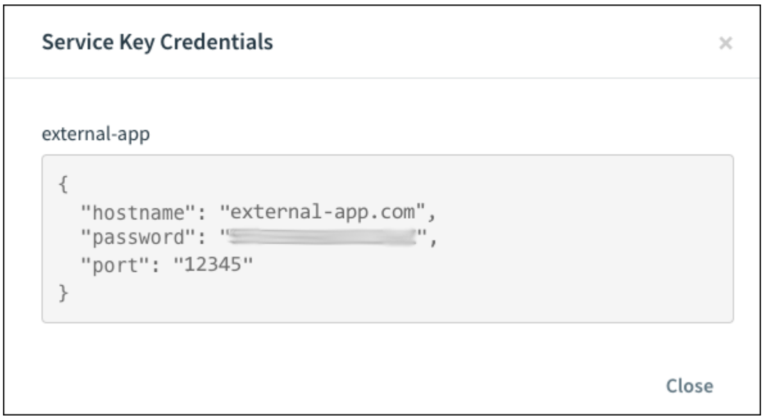
3. (Optional) Click **Show Advanced Options**. Under **Arbitrary Parameters**, enter any additional service-specific configuration in the **Name** and **Value** fields.

4. Click **Create** to generate the service key.

View Credentials for a Service Key

Follow the steps below to view the credentials for a service key.

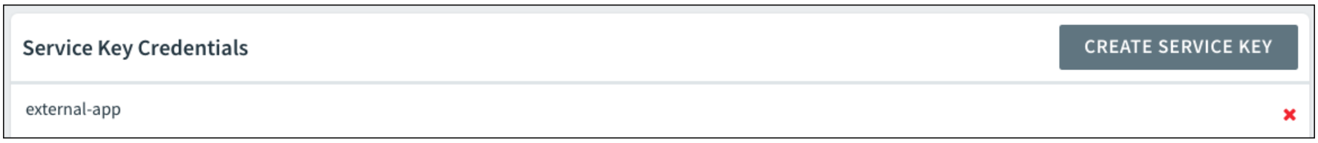
1. To view the credentials for a particular service instance, click the service instance name under **Service Key Credentials**. The JSON object containing the credentials appears.



2. Click **Close**.

Delete Service Key

To delete a service key, click the red `x` next to the service instance name.



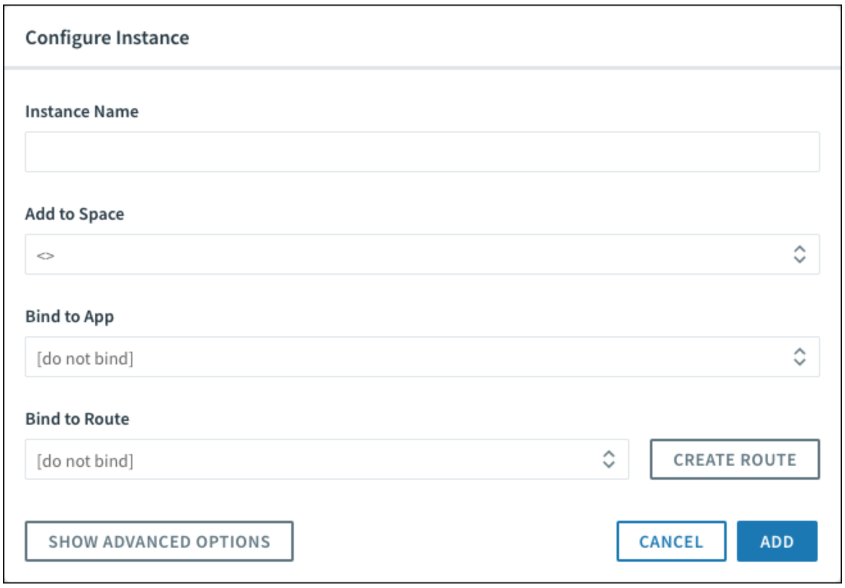
Manage Route Services

For more information about route services, see the Route Services topic.

You can bind a new service instance to a route when creating the instance in the Marketplace, or you can manage route services for an existing service instance on the service instance page.

Bind a New Service Instance to a Route

Follow the steps below to bind a new service instance to a route.



1. Select the service from the Marketplace.
2. Under **Bind to Route**, either bind the service instance to an existing route or click **Create Route** to create a new custom route.

   > **Note**: You must choose a Marketplace service compatible with route services for the **Bind to Route** field to appear.

3. Complete the remaining fields and click **Add** to create the service instance.

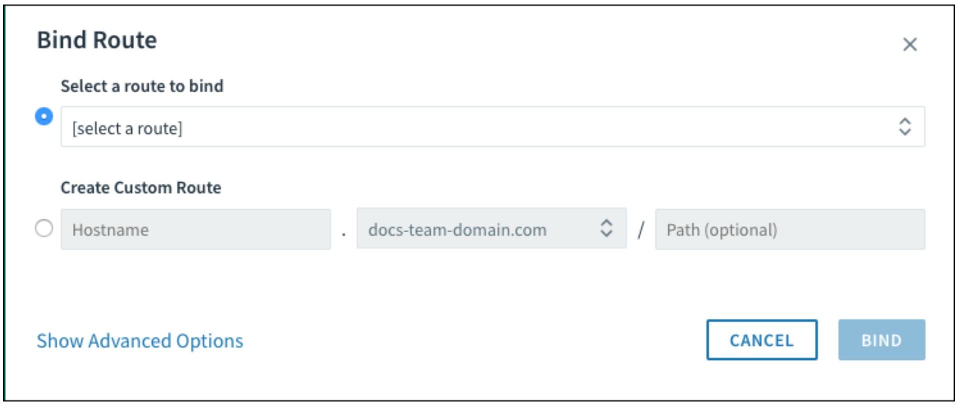
Bind an Existing Service Instance to a Route

Follow the steps below to bind an existing service instance to a route.

1. On the space page, click **Services**.
2. Click the service instance that you want to manage route services for.

   > **Note**: If the service is not compatible with route services, the text "This service does not support route binding" appears under **Bound Routes**.

3. To bind the service instance to a route, click **Bind Route**.



4. Select an existing route under **Select a route to bind** or enter a new route under **Create Custom Route**.

5. Click **Bind**.

To unbind a route from a service instance, click the red `x` next to the name of the route under **Bound Routes**.

Viewing ASGs in Apps Manager

Page last updated:

About ASGs

Application Security Groups (ASGs) are a collections of egress rules that specify the protocols, ports, and IP address ranges where app or task instances send traffic. The platform sets up rules to filter and log outbound network traffic from app and task instances. ASGs apply to both buildpack-based and Docker-based apps and tasks.

When apps or tasks begin staging, they need traffic rules permissive enough to allow them to pull resources from the network. After an app or task is running, the traffic rules can be more restrictive and secure. To distinguish between these two security requirements, administrators can define one ASG for app and task staging, and another for app and task runtime. For more information about staging and running apps, see Application Container Lifecycle.

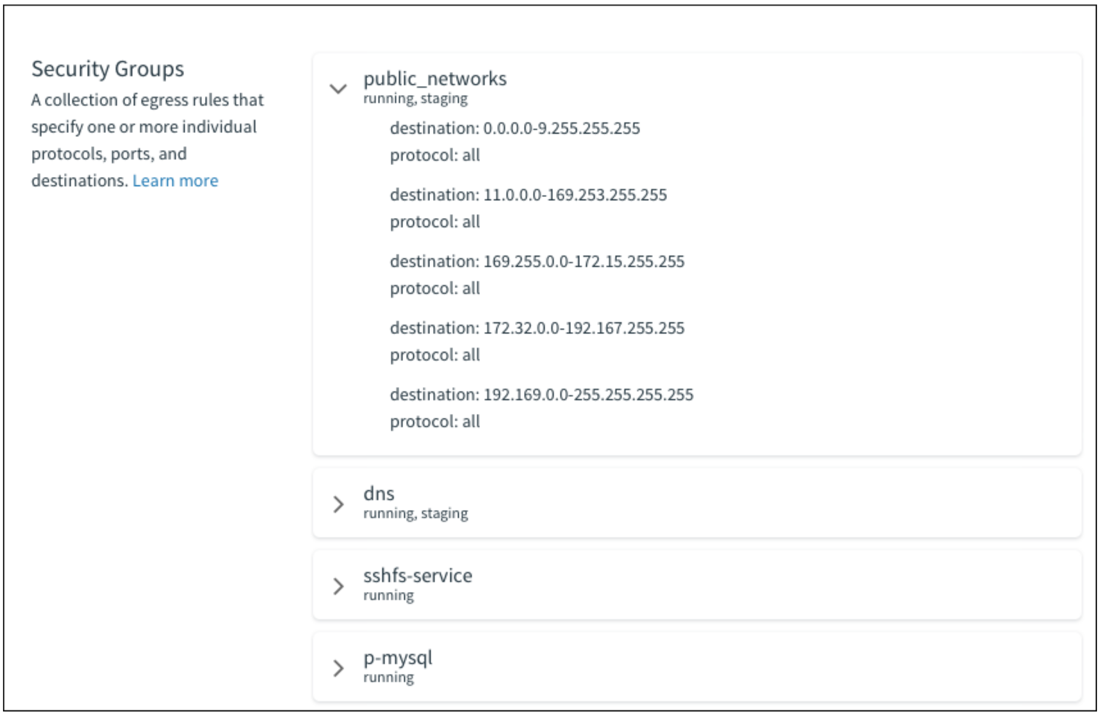
To provide granular control when securing a deployment, an administrator can assign ASGs to apply to all app and task instances for the entire deployment, or assign ASGs to spaces to apply only to apps and tasks in a particular space.

Only admin users can create and modify ASGs. For information about creating and configuring ASGs, see Application Security Groups.

Displaying ASGs for a Space

To view the ASGs associated with a space, perform the following steps.

1. Log in to Apps Manager.
2. From the **Org** dropdown, select the **Org** that contains the space you want to view.
3. Select the **Space** you want to view.
4. Click on the **Settings** tab.
5. In the **Security Groups** section, Apps Manager displays ASGs associated with the selected space.
6. Click on an ASG to expand its egress rules.



Configuring Spring Boot Actuator Endpoints for Apps Manager

Page last updated:

The Apps Manager UI supports several production-ready endpoints from Spring Boot Actuator. This topic describes the Actuator endpoints and how you can configure your app to display data from the endpoints in Apps Manager.

For more information about Spring Boot Actuator, see the Spring Boot Actuator documentation.

> **Note**: This feature requires Spring Boot v1.5 or later.

Overview

The Apps Manager integration with Spring Boot does not use the standard Spring Boot Actuators. Instead, it uses a specific set of actuators that are secured using the Space Developer role for the space that the application runs in. Authentication and authorization are automatically delegated to the Cloud Controller and the User Account and Authentication server without any configuration from the user.

By default, actuators are secure and cannot be accessed without explicit configuration by the user, even if Spring Security is not included. This allows users to take advantage of the Spring Boot Apps Manager integration without accidentally exposing their actuators without security.

Actuator Endpoints

The table below describes the Spring Boot Actuator endpoints supported by Apps Manager. To integrate these endpoints with Apps Manager, you must first Activate Spring Boot Actuator for Your App.

Endpoint	About
`/info`	• **Description**: Exposes details about app environment, git, and build. To send build and Git information to this endpoint, see Configure the Info Actuator. • **How to use in Apps Man**: See View Build and Git Information for Your App.
`/health`	• **Description**: Shows health status or detailed health information over a secure connection. Spring Boot Actuator includes the auto-configured health indicators specified in the Auto-configured HealthIndicators section of the Spring Boot documentation. If you want to write custom health indicators, see the Writing custom HealthIndicators section of the Spring Boot documentation. • **How to use in Apps Man**: See View App Health.
`/loggers`	• **Description**: Lists and allows modification of the levels of the loggers in an app. • **How to use in Apps Man**: See Manage Log Levels.
`/dump`	• **Description**: Generates a thread dump. • **How to use in Apps Man**: See View Thread Dump.
`/trace`	• **Description**: Displays trace information from your app for each of the last 100 HTTP requests. For more information, see the Tracing section of the Spring Boot documentation. • **How to use in Apps Man**: See View Request Traces.
`/heapdump`	• **Description**: Generates a heap dump and provides a compressed file containing the results. • **How to use in Apps Man**: See Download Heap Dump.
`/mappings`	• **Description**: Displays the endpoints an app serves and other related details. • **How to use in Apps Man**: See View Mappings.

Activate Spring Boot Actuator for Your App

You must add a `spring-boot-starter-actuator` dependency to your app project for the production-ready HTTP endpoints to return values. For more information, see the Enabling production-ready features section of the Spring Boot documentation.

1. Follow the instructions below that correspond to your project type.
   - **Maven**: If you use Maven, add the following to your project:

     ```
     <dependencies>
         <dependency>
             <groupId>org.springframework.boot</groupId>
             <artifactId>spring-boot-starter-actuator</artifactId>
         </dependency>
     </dependencies>
     ```

   - **Gradle**: If you use Gradle, add the following to your project:

     ```
     dependencies {
         compile("org.springframework.boot:spring-boot-starter-actuator")
     }
     ```

2. If you use self-signed certificates in your PCF deployment for UAA or the Cloud Controller, specify in your `application.properties` file to skip SSL validation:

   ```
   management.cloudfoundry.skip-ssl-validation=true
   ```

   See Cloud Foundry support in the Spring Boot Actuator documentation for more information.

Configure the Info Actuator

The `/info` endpoint provides information about the project build for your app, as well as its git details.

Add Build Information

To add build information to the `/info` endpoint, follow the instructions below that correspond to your project type.

Maven

Add the following to your app project:

```
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
      <version>1.4.2.RELEASE</version>
      <executions>
        <execution>
          <goals>
            <goal>build-info</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
```

Gradle

Add the following to your app project:

```
springBoot {
  buildInfo()
}
```

Add Git Information

To add git information to the `/info` endpoint, follow these instructions:

1. Add the following property to your `application.properties` file:

```
management.info.git.mode=full
```

2. Follow the instructions below that correspond to your project type.

Maven

Add the following plugin to your project:

```
<build>
    <plugins>
        <plugin>
            <groupId>pl.project13.maven</groupId>
            <artifactId>git-commit-id-plugin</artifactId>
        </plugin>
    </plugins>
</build>
```

Gradle

Add the following plugin to your project:

```
plugins {
    id "com.gorylenko.gradle-git-properties" version "1.4.17"
}
```

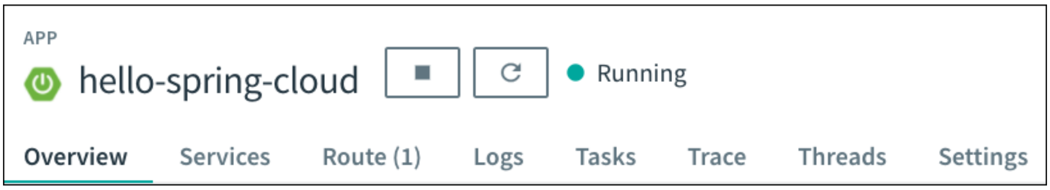
Using Spring Boot Actuators with Apps Manager

This document describes how to view and manage app information from Spring Boot Actuator in Apps Manager.

Prerequisites

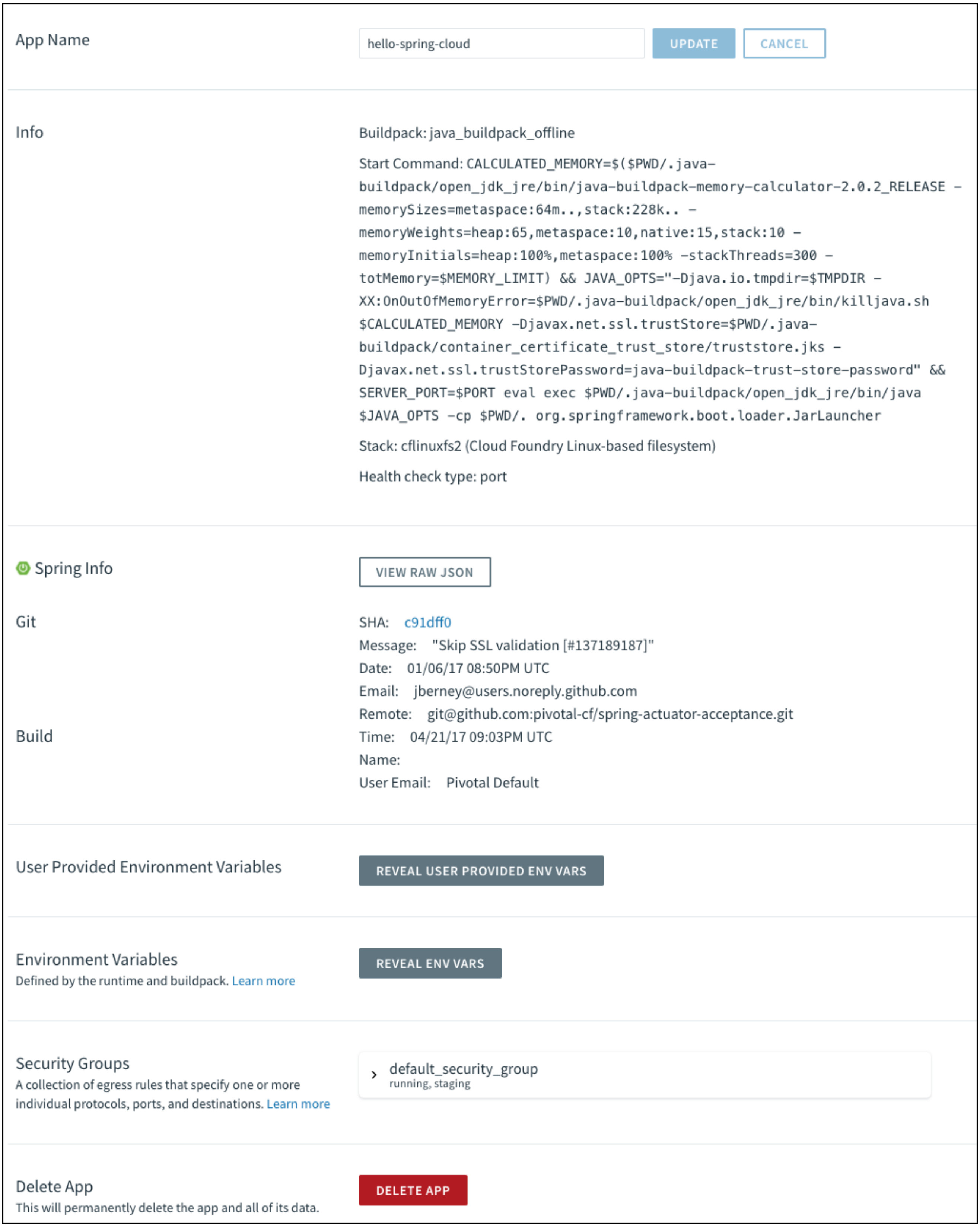
The Apps Manager integration with Spring Boot Actuator requires the following:

- A PCF user with the `SpaceDeveloper` role. See App Space Roles.
- Spring Boot v1.5 or later.
- Completing the procedures in Configure Spring Boot Actuator Endpoints for Apps Manager.
  After your configure your app, Apps Manager displays the Spring Boot logo next to the name of your app on the app page:



View Build and Git Information for Your App

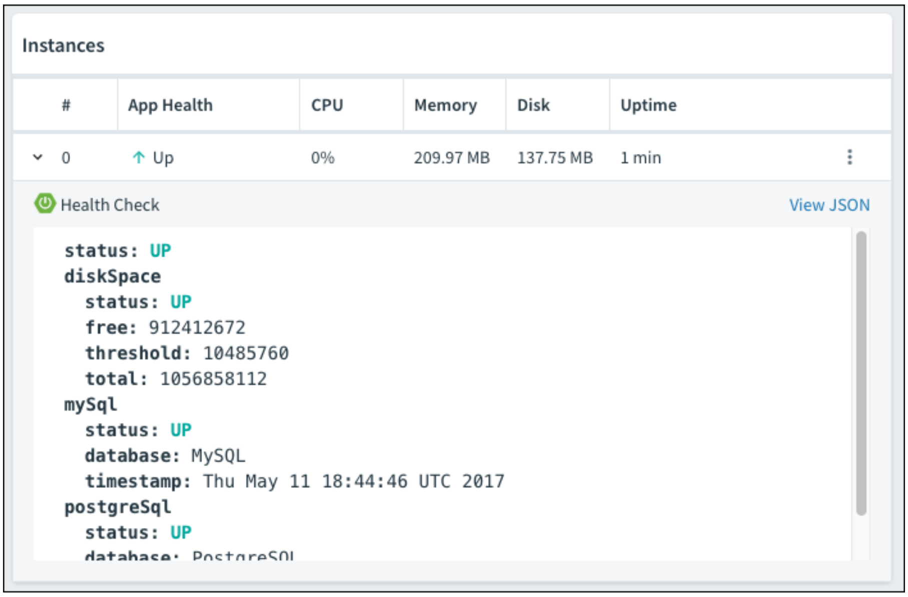
To view the data that your app sends to its `/info` Actuator endpoint, select the **Settings** tab:



In the upper right of the app page, Apps Manager also displays the SHA of your app code repository from the latest build:

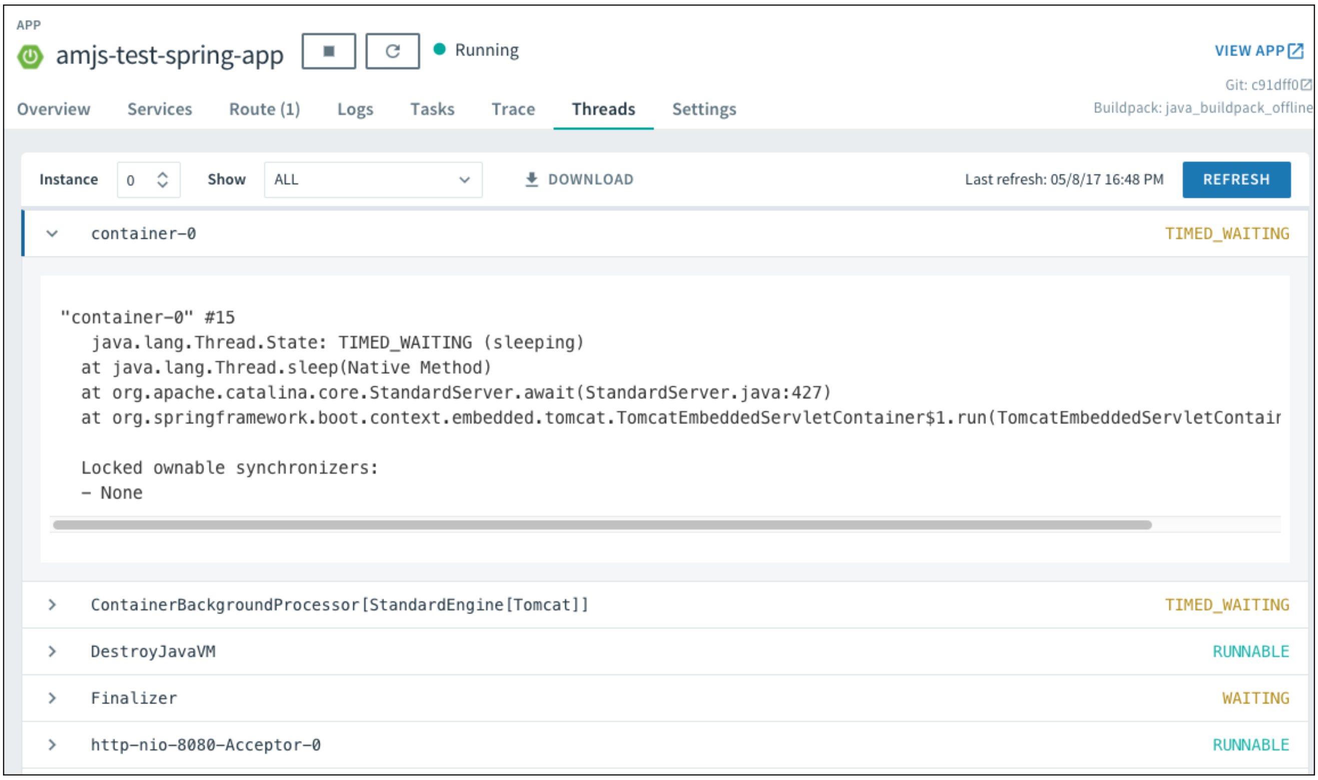
View App Health

To view the health-check data that your app sends to its `/health` Actuator endpoints, select the **Overview** tab and click an instance under the **Instances** section:



View Thread Dump

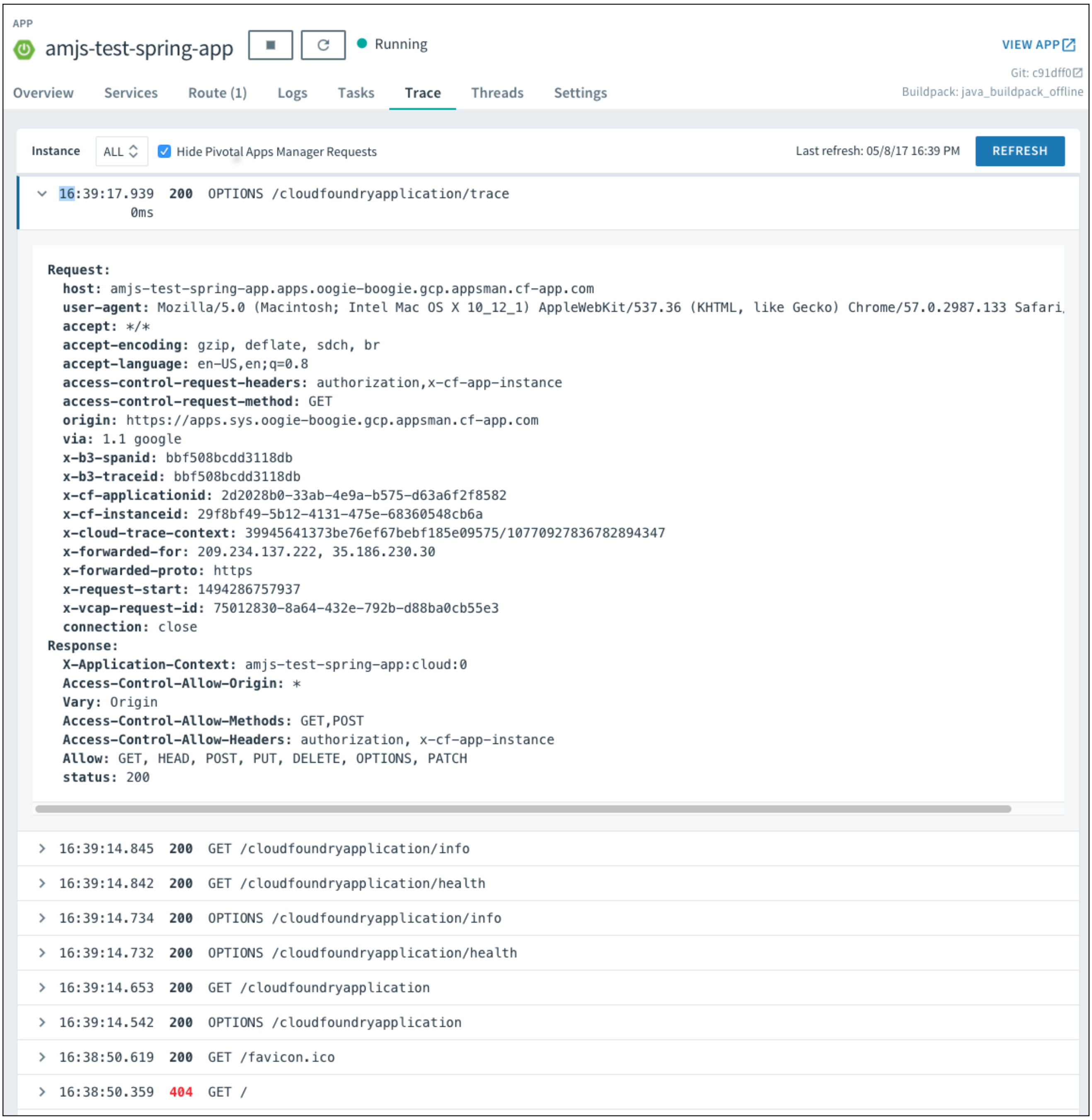
To trigger and view a thread dump from your app to its `/dump` Actuator endpoint, select the **Threads** tab and click **Refresh**.



You can click each thread to expand and view its details. You can also modify which threads appear on the page using the **Instance** and **Show** drop-down menus.

View Request Traces

To retrieve and view tracing information from the `/trace` Actuator endpoint of your app, select the **Trace** tab and click **Refresh**.

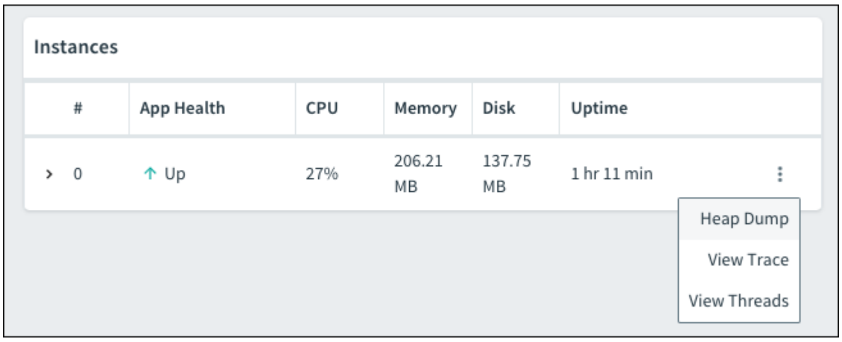


This page displays the last 100 requests from your app. You can click each individual request to expand and view its trace details. You can modify which requests appear on the page using the **Instance** drop-down menu.

By default, the **Trace** tab does not show requests and responses from Apps Manager polling app instances for data. To include these requests, clear the **Hide Pivotal Apps Manager Requests** checkbox next to the **Instance** drop-down menu.

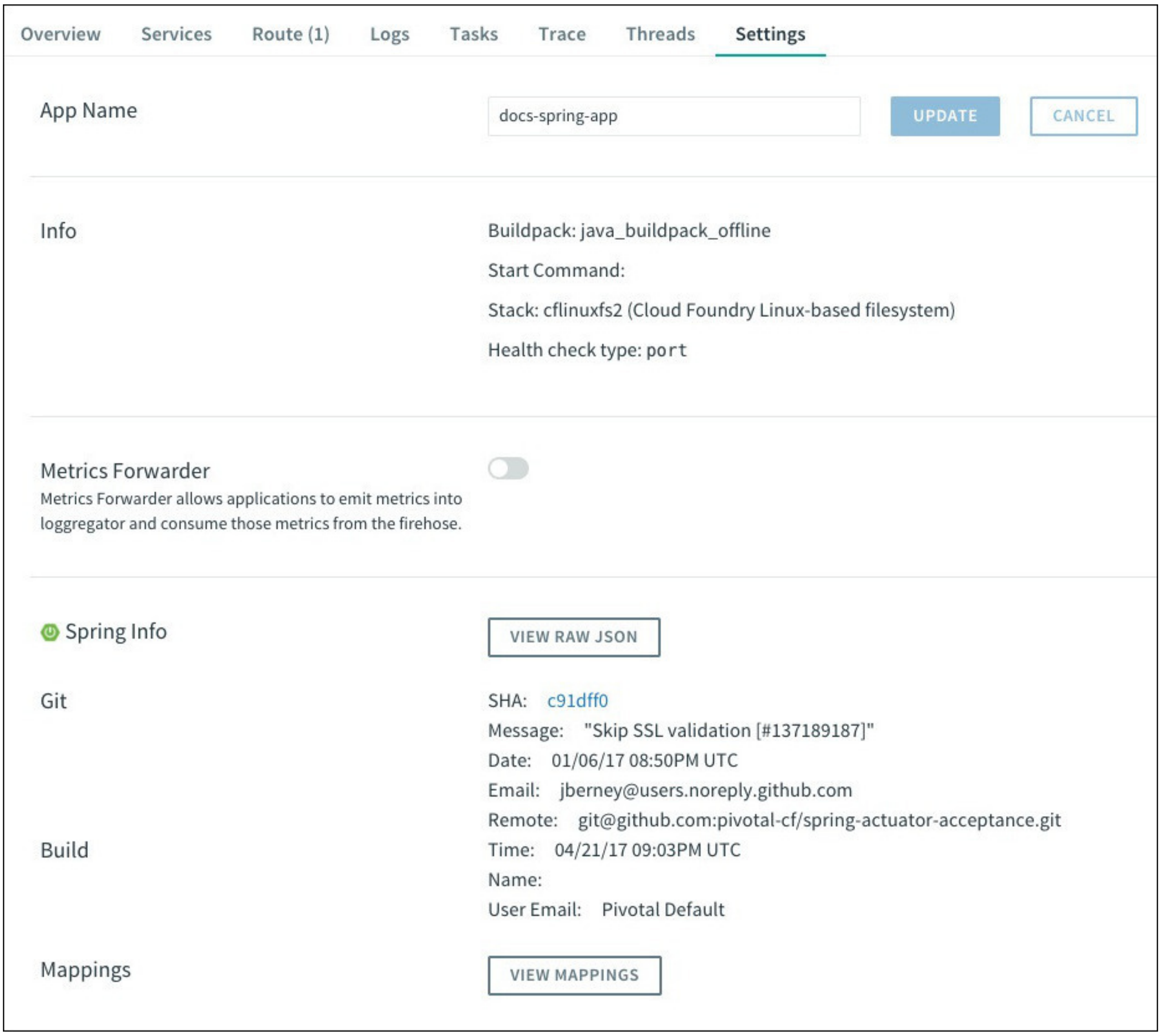
Download Heap Dump

To trigger and view a heap dump from your app to its `/heapdump` endpoint, select the settings drop-down menu for an instance of your app and click **Heap Dump**. This downloads a `.zip` file.



View Mappings

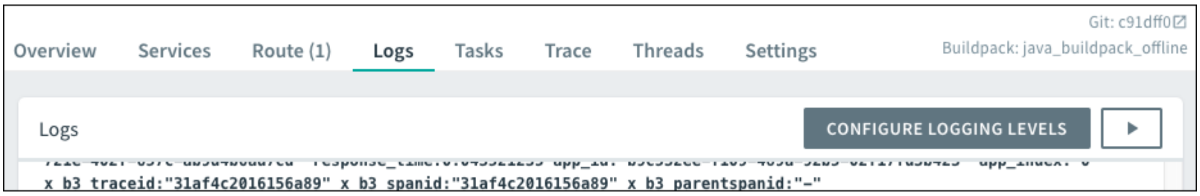
To view a collated list of the endpoints an app serves, select the **Settings** tab and click **View Mappings**.



Manage Log Levels

Spring Boot apps include *loggers* for many provided and user components of the app. You can set the log level for each logger in Apps Manager.

To view the **Configure Logging Levels** screen, select the **Logs** tab and click **Configure Logging Levels**.



Apps Manager displays the default log level for each logger in gray.

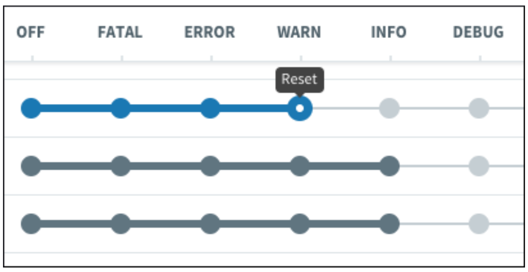
You can modify the log level for a logger by clicking the desired level in the logger row, as shown in the image below. Whenever you set a log level, the following happens:

- The log level displays in blue to indicate that it is user-configured.
- Each child namespace of the logger inherits the log level.

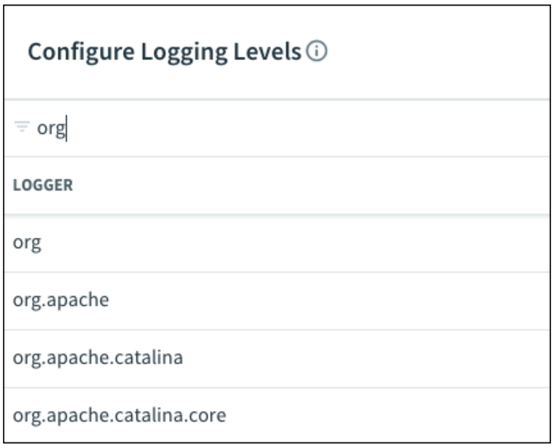
> **Note**: You can manually set any of the child loggers to override this inheritance.

All of the loggers with user-configured logging levels float to the top of the list.

You can reset log levels by clicking the white dot displayed on the current log level.



You can also filter which loggers you see using the **Filter Loggers** textbox.



Troubleshoot Spring Boot Actuator Integration

This section describes how to troubleshoot common issues with the integration of Apps Manager and Spring Boot Actuator.

/cloudfoundryapplication Failed Request

Symptom

You see the following failed request message in your app logs:

```
Could not find resource for relative : /cloudfoundryapplication of full path: http://example.com/cloudfoundryapplication
```

Explanation

Apps Manager uses the `/cloudfoundryapplication` endpoint as the root for Spring Boot Actuator integrations. It calls this endpoint for an app when you view the app in the Apps Manager UI, regardless of whether you have configured Spring Boot Actuator endpoints for Apps Manager.

Solution

If you are not using the Spring Boot Actuator integrations for Apps Manager, you can ignore this failed request message.

Cloud Foundry Command Line Interface (cf CLI)

This guide explains the Cloud Foundry Command Line Interface (cf CLI), a tool you use to deploy and manage your applications.

Contents in this section:

Installing the cf CLI

Page last updated:

This topic describes how to install the Cloud Foundry Command Line Interface (cf CLI). Follow the instructions below for your operating system. If you previously used the cf CLI v5 Ruby gem, uninstall this gem first.

You can install the cf CLI with a package manager, an installer, or a compressed binary.

Note: For use with Pivotal Cloud Foundry v1.10, the recommended minimum version is cf CLI v6.23 or later.

Use a Package Manager

Mac OS X Installation

For Mac OS X, perform the following steps to install the cf CLI with Homebrew:

1. Tap the Cloud Foundry formula repository:

```
$ brew tap cloudfoundry/tap
```

2. Install the cf CLI:

```
$ brew install cf-cli
```

Linux Installation

For Debian and Ubuntu-based Linux distributions, perform the following steps:

1. Add the Cloud Foundry Foundation public key and package repository to your system:

```
$ wget -q -O - https://packages.cloudfoundry.org/debian/cli.cloudfoundry.org.key | sudo apt-key add -
```

```
$ echo "deb http://packages.cloudfoundry.org/debian stable main" | sudo tee /etc/apt/sources.list.d/cloudfoundry-cli.list
```

2. Update your local package index:

```
$ sudo apt-get update
```

3. Install the cf CLI:

```
$ sudo apt-get install cf-cli
```

For Enterprise Linux and Fedora systems (RHEL6/CentOS6 and up), perform the following steps:

1. Configure the Cloud Foundry Foundation package repository:

```
$ sudo wget -O /etc/yum.repos.d/cloudfoundry-cli.repo https://packages.cloudfoundry.org/fedora/cloudfoundry-cli.repo
```

2. Install the cf CLI, which also downloads and adds the public key to your system:

```
$ sudo yum install cf-cli
```

Use an Installer

Follow the instructions for your operating system below.

Windows Installation

To use the cf CLI installer for Windows, perform the following steps:

1. Download the Windows installer.
2. Unpack the zip file.
3. Double click the `cf CLI` executable.
4. When prompted, click **Install**, then **Close**.
5. To verify your installation, open a terminal window and type `cf`. If your installation was successful, the cf CLI help listing appears.

Mac OS X Installation

To use the cf CLI installer for Mac OS X, perform the following steps:

1. Download the OS X installer.
2. Open the `.pkg` file.
3. In the installer wizard, click **Continue**.
4. Select an install destination and click **Continue**.
5. When prompted, click **Install**.
6. To verify your installation, open a terminal window and type `cf`. If your installation was successful, the cf CLI help listing appears.

Linux Installation

To use the cf CLI installer for Linux, perform the following steps:

1. Download the Linux installer for your Debian/Ubuntu or Red Hat system.
2. Install using your system's package manager. Note these commands may require `sudo`.
   - For Debian/Ubuntu, run the following command:

     ```
     $ dpkg -i path/to/cf-cli-*.deb && apt-get install -f
     ```

   - For Red Hat, run the following command:

     ```
     rpm -i path/to/cf-cli-*.rpm
     ```

3. To verify your installation, open a terminal window and type `cf`. If your installation was successful, the cf CLI help listing appears.

Use a Compressed Binary

Download the compressed binary for Mac OS X, Windows, or Linux from the cf CLI GitHub repository and install it on your system.

The specific procedures vary by operating system, but the following example illustrates downloading and installing the binary on Mac OS X:

1. Download and extract the Mac OS X binary:

```
$ curl -L "https://cli.run.pivotal.io/stable?release=macosx64-binary&source=github" | tar -zx
```

2. Move it to `/usr/local/bin`, or another location in your `$PATH`:

```
$ mv cf /usr/local/bin
```

3. Confirm your cf CLI version:

```
$ cf --version
```

Next Steps

See Getting Started with cf CLI for more information about how to use the cf CLI.

We recommend that you review our CLI releases page to learn when updates are released, and download a new binary or a new installer when you want to update to the latest version.

Uninstall the cf CLI

Package Manager

If you previously installed the cf CLI with a package manager, follow the instructions specific to your package manager to uninstall the cf CLI.

The specific procedures vary by package manager, but the following example illustrates uninstalling the cf CLI with Homebrew:

```
$ brew uninstall cf-cli
```

Installer

If you previously installed the cf CLI with an installer, perform the instructions specific to your operating system to uninstall the cf CLI:

- For Mac OS, delete the binary `/usr/local/bin/cf`, and the directory `/usr/local/share/doc/cf-cli`.
- For Windows, navigate to the **Control Panel**, click **Programs and Features**, select `Cloud Foundry CLI VERSION` and click **Uninstall**.

Binary

If you previously installed a cf CLI binary, remove the binary from where you copied it.

cf CLI v5

To uninstall, run `gem uninstall cf`.

Note: To ensure that your Ruby environment manager registers the change, close and reopen your terminal.

Getting Started with the cf CLI

Page last updated:

This topic describes configuring and getting started with the Cloud Foundry Command Line Interface (cf CLI). This page assumes you have the latest version of the cf CLI. See the Installing the Cloud Foundry Command Line Interface topic for installation instructions.

Localize

The cf CLI translates terminal output into the language that you select. The default language is `en-US`. The cf CLI supports the following languages:

- Chinese (simplified): `zh-Hans`
- Chinese (traditional): `zh-Hant`
- English: `en-US`
- French: `fr-FR`
- German: `de-DE`
- Italian: `it-IT`
- Japanese: `ja-JP`
- Korean: `ko-KR`
- Portuguese (Brazil): `pt-BR`
- Spanish: `es-ES`

Use cf config to set the language. To set the language with `cf config`, use the syntax: `$ cf config --locale YOUR_LANGUAGE`.

For example, to set the language to Portuguese and confirm the change by running `cf help`:

```
$ cf config --locale pt-BR
$ cf help
NOME:
   cf - Uma ferramenta de linha de comando para interagir com Cloud Foundry

USO:
   cf [opções globais] comando [argumentos...] [opções de comando]

VERSÃO:
   6.14.1+dc6adf6-2015-12-22
   ...
```

Note: Localization with `cf config --locale` affects only messages that the cf CLI generates.

Login

Use cf login to log in to PAS. The `cf login` command uses the following syntax to specify a target API endpoint, an org (organization), and a space: `$ cf login [-a API_URL] [-u USERNAME] [-p PASSWORD] [-o ORG] [-s SPACE]`.

- `API_URL`: This is your API endpoint, the URL of the Cloud Controller in your PAS instance.
- `USERNAME`: Your username.
- `PASSWORD`: Your password. Use of the `-p` option is discouraged as it may record your password in your shell history.
- `ORG`: The org where you want to deploy your apps.
- `SPACE`: The space in the org where you want to deploy your apps.

The cf CLI prompts for credentials as needed. If you are a member of multiple orgs or spaces, `cf login` prompts you for which ones to log into. Otherwise it targets your org and space automatically.

```
$ cf login -a https://api.example.com -u username@example.com
API endpoint: https://api.example.com

Password>
Authenticating...
OK

Select an org (or press enter to skip):
1. example-org
2. example-other-org

Org> 1
Targeted org example-org

Select a space (or press enter to skip):
1. development
2. staging
3. production

Space> 1
Targeted space development
```

Alternatively, you can write a script to log in and set your target using the non-interactive cf api, cf auth, and cf target commands.

Upon successful login, the cf CLI saves a `config.json` file containing your API endpoint, org, space values, and access token. If you change these settings, the `config.json` file is updated accordingly.

By default, `config.json` is located in your `~/.cf` directory. The `CF_HOME` environment variable allows you to locate the `config.json` file wherever you like.

Users and Roles

The cf CLI includes commands that list users and assign roles in orgs and spaces. See the Orgs, Spaces, Roles, and Permissions topic.

Commands for Listing Users

These commands take an org or space as an argument:

- cf org-users
- cf space-users

For example, to list the users who are members of an org:

```
$ cf org-users example-org
Getting users in org example-org as username@example.com...

ORG MANAGER
  username@example.com

BILLING MANAGER
  huey@example.com
  dewey@example.com

ORG AUDITOR
  louie@example.com
```

Commands for Managing Roles

These commands require PAS admin permissions and take username, org or space, and role as arguments:

- cf set-org-role
- cf unset-org-role
- cf set-space-role
- cf unset-space-role

Available roles are “OrgManager”, “BillingManager”, “OrgAuditor”, “SpaceManager”, “SpaceDeveloper”, and “SpaceAuditor”. For example, to grant the

Org Manager role to a user within an org:

```
$ cf set-org-role huey@example.com example-org OrgManager

Assigning role OrgManager to user huey@example.com in org example-org as username@example.com...
OK
```

Note: If you are not a PAS admin, you see this message when you try to run these commands: `error code: 10003, message: You are not authorized to perform the requested action`

Push

The cf push command pushes a new app or syncs changes to an existing app.

If you do not provide a hostname (also known as subdomain), `cf push` routes your app to a URL of the form `APPNAME.DOMAIN` based on the name of your app and your default domain. If you want to map a different route to your app, see the Routes and Domains topic for information about creating routes.

The `cf push` command supports many options that determine how and where the app instances are deployed. For details about the `cf push` command, see the push page in the Cloud Foundry CLI Reference Guide.

The following example pushes an app called `my-awesome-app` to the URL `http://my-awesome-app.example.com` and specifies the Ruby buildpack with the `-b` flag.

Note: When you push an app and specify a buildpack with the `-b` flag, the app remains permanently linked to that buildpack. To use the app with a different buildpack, you must delete the app and re-push it.

```
$ cf push my-awesome-app -b ruby_buildpack
Creating app my-awesome-app in org example-org / space development as username@example.com...
OK

Creating route my-awesome-app.example.com...
OK
...

1 of 1 instances running

App started
...

requested state: started
instances: 1/1
usage: 1G x 1 instances
urls: my-awesome-app.example.com
last uploaded: Wed Jun 8 23:43:15 UTC 2016
stack: cflinuxfs2
buildpack: ruby_buildpack

     state     since                    cpu    memory    disk      details
#0   running   2016-06-08 04:44:07 PM   0.0%   0 of 1G   0 of 1G
```

For more information about available buildpacks, see the Buildpacks topic.

User-Provided Service Instances

To create or update a user-provided service instance, you need to supply basic parameters. For example a database service might require a username, password, host, port, and database name.

The cf CLI has three ways of supplying these parameters to create or update an instance of a service: interactively, non-interactively, and in conjunction with third-party log management software as described in RFC 6587. When used with third-party logging, the cf CLI sends data formatted according to RFC 5424.

You create a service instance with `cf cups` and update one with `cf uups` as described below.

The cf create-user-provided-service (cups) Command

Use cf create-user-provided-service (alias `cf cups`) creates a new service instance.

To supply service instance parameters interactively: Specify parameters in a comma-separated list after the `-p` flag. This example command-line session creates a service instance for a database service.

```
$ cf cups sql-service-instance -p "host, port, dbname, username, password"
host> mysql.example.com
port> 1433
dbname> mysqldb
username> admin
password> Pa55w0rd
Creating user provided service sql-service-instance in org example-org / space development as username@example.com...
OK
```

To supply service instance parameters to `cf cups` non-interactively: Pass parameters and their values in as a JSON hash, bound by single quotes, after the `-p` tag. This example is a non-interactive version of the `cf cups` session above.

```
$ cf cups sql-service-instance -p '{"host":"mysql.example.com", "port":"1433", "dbname":"mysqldb", "username":"admin","password":"pa55woRD"}'
Creating user provided service sql-service-instance in org example-org / space development as username@example.com...
OK
```

To create a service instance that sends data to a third-party: Use the `-l` option followed by the external destination URL. This example creates a service instance that sends log information to the syslog drain URL of a third-party log management service. For specific log service instructions, see the Service-Specific Instructions for Streaming Application Logs topic.

```
$ cf cups mylog -l syslog://logs4.example.com:25258
Creating user provided service mylog in org example-org / space development as username@example.com...
OK
```

After you create a user-provided service instance, you bind it to an app with cf bind-service, unbind it with cf unbind-service, rename it with cf rename-service, and delete it with cf delete-service.

The cf update-user-provided-service (uups) Command

Use cf update-user-provided-service (alias `cf uups`) to update one or more of the parameters for an existing user-provided service instance. The `cf uups` command uses the same syntax as `cf cups` above to set parameter values. The `cf uups` command does not update any parameter values that you do not supply.

cf CLI Return Codes

The cf CLI uses exit codes, which help with scripting and confirming that a command has run successfully. For example, after you run a cf CLI command, you can retrieve its return code by running `echo $?` (on Windows, `echo %ERRORLEVEL%`). If the return code is `0`, the command was successful.

The cf help Command

The cf help command lists the cf CLI commands and a brief description of each. Passing the `-h` flag to any command lists detailed help, including any aliases. For example, to see detailed help for `cf delete`, run:

```
$ cf delete -h
NAME:
   delete - Delete an app

USAGE:
   cf delete APP_NAME [-f -r]

ALIAS:
   d

OPTIONS:
   -f      Force deletion without confirmation
   -r      Also delete any mapped routes
```

Using the cf CLI with a Proxy Server

Page last updated:

If you have an HTTP or SOCKS5 proxy server on your network between a host running the cf CLI and your Cloud Foundry API endpoint, you must set `https_proxy` with the hostname or IP address of the proxy server.

The `https_proxy` environment variable holds the hostname or IP address of your proxy server.

`https_proxy` is a standard environment variable. Like any environment variable, the specific steps you use to set it depends on your operating system.

Format of https_proxy

`https_proxy` is set with hostname or IP address of the proxy server in URL format: `https_proxy=http://proxy.example.com`

If the proxy server requires a user name and password, include the credentials: `https_proxy=http://username:password@proxy.example.com`

If the proxy server uses a port other than 80, include the port number: `https_proxy=http://username:password@proxy.example.com:8080`

If the proxy server is a SOCKS5 proxy, specify the SOCKS5 protocol in the URL: `https_proxy=socks5://socks_proxy.example.com`

Note: `cf ssh` does not work through a SOCKS5 proxy.

Setting https_proxy in Mac OS or Linux

Set the `https_proxy` environment variable using the command specific to your shell. For example, in bash, use the `export` command.

Example:

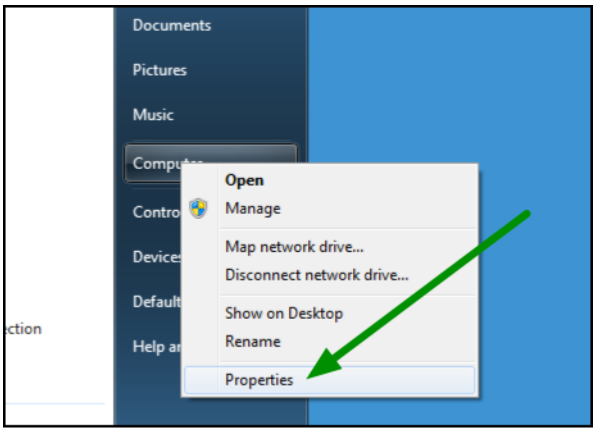
```
$ export https_proxy=http://my.proxyserver.com:8080
```

To make this change persistent, add the command to the appropriate profile file for the shell. For example, in bash, add a line like the following to your `.bash_profile` or `.bashrc` file:

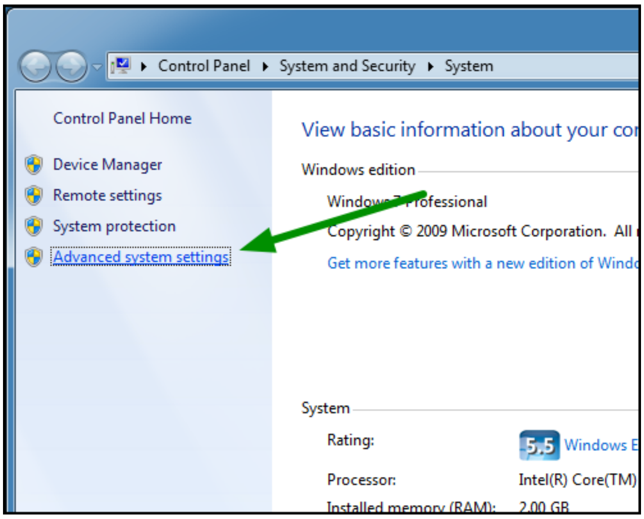
```
https_proxy=http://username:password@hostname:port
export $https_proxy
```

Setting https_proxy in Windows

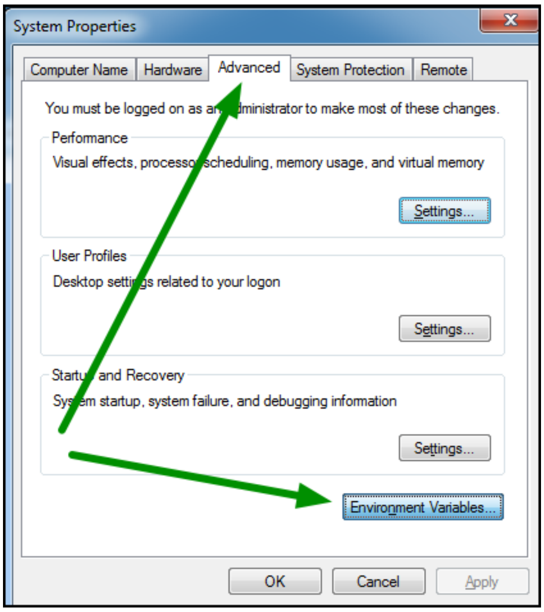
1. Open the Start menu. Right-click **Computer** and select **Properties**.



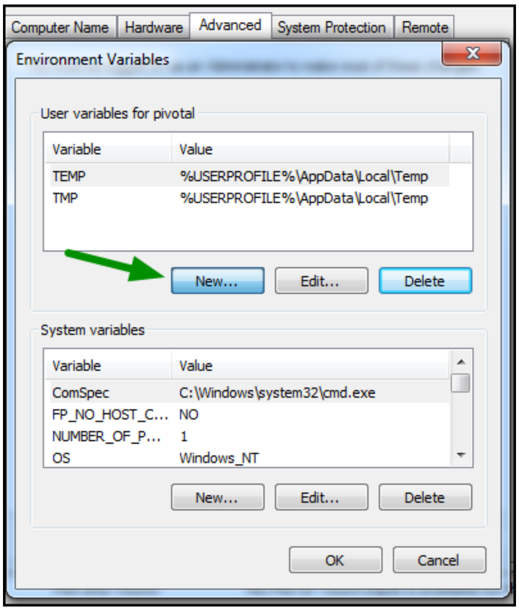
2. In the left pane of the System window, click **Advanced system settings**.



3. In the System Properties window, select the **Advanced** tab, then click **Environment Variables**.



4. In the Environment Variables window, under User variables, click **New**.



5. In the Variable name field, input `https_proxy`. In the Variable value field, input your proxy server information.

6. Click **OK**.

Using the cf CLI with a Self-Signed Certificate

Page last updated:

This topic describes how developers can use the cf CLI to communicate securely with a Pivotal Cloud Foundry (PCF) deployment without specifying `--skip-ssl-validation` under the following circumstances:

- The deployment uses a self-signed certificate.
- The deployment uses a certificate that is signed by a self-signed certificate authority (CA), or a certificate signed by a certificate that's signed by a self-signed CA.

Before following the procedure below, the developer must obtain either the self-signed certificate or the intermediate and CA certificate(s) used to sign the deployment's certificate. The developer can obtain these certificates from the PCF operator.

Install the Certificate on Local Machines

The certificates that developers must insert into their local truststore vary depending on the configuration of the deployment.

- If the deployment uses a self-signed certificate, the developer must insert the self-signed certificate into their local truststore.
- If the deployment uses a certificate that is signed by a self-signed certificate authority (CA), or a certificate signed by a certificate that's signed by a self-signed CA, the developer must insert the self-signed certificate and any intermediate certificates into their local truststore.

Installing the Certificate on Mac OS X

Enter the following command to place a certificate file `server.crt` into your local truststore:

```
$ sudo security add-trusted-cert -d -r trustRoot -k /Library/Keychains/System.keychain server.crt
```

Installing the Certificate on Linux

Perform the following steps specific to your distribution to place the certificate file `server.crt` into your truststore:

- Debian/Ubuntu/Gentoo:

  ```
  $ cat server.crt >> /etc/ssl/certs/ca-certificates.crt
  ```

- Fedora/RHEL:

  ```
  $ cat server.crt >> /etc/pki/tls/certs/ca-bundle.crt
  ```

The above example will set certificate permanently on your machine accross all users and requires sudo permissions. You can also run the following command to set certificate in your current terminal/script:

```
$ export SSL_CERT_FILE=/path/to/server.crt
```

or

```
$ export SSL_CERT_DIR=/path/to/server/dir
```

Installing the Certificate on Windows

1. Right-click on the certificate file and click **Install Certificate**.
2. Choose to install the certificate as the **Current User** or **Local Machine**. Choose the **Trusted Root Certification Authorities** as the certification store.

Using cf CLI Plugins

Page last updated:

The Cloud Foundry Command Line Interface (cf CLI) includes plugin functionality. These plugins enable developers to add custom commands to the cf CLI. You can install and use plugins that Cloud Foundry developers and third-party developers create. You can review the Cloud Foundry Community CLI Plugin page for a current list of community-supported plugins. You can find information about submitting your own plugin to the community in the Cloud Foundry CLI plugin repository on GitHub.

Warning: Plugins are not vetted in any way, including for security or functionality. Use plugins at your own risk.

The cf CLI identifies a plugin by its binary filename, its developer-defined plugin name, and the commands that the plugin provides. You use the binary filename only to install a plugin. You use the plugin name or a command for any other action.

Note: The cf CLI uses case-sensitive commands, but plugin management commands accept plugin and repository names irrespective of their casing.

Changing the Plugin Directory

By default, the cf CLI stores plugins on your workstation in `$CF_HOME/.cf/plugins`, which defaults to `$HOME/.cf/plugins`. To change the root directory of this path from `$CF_HOME`, set the `CF_PLUGIN_HOME` environment variable. The cf CLI appends `.cf/plugins` to the `CF_PLUGIN_HOME` path that you specify and stores plugins in that location.

For example, if you set `CF_PLUGIN_HOME` to `/my-folder`, cf CLI stores plugins in `/my-folder/.cf/plugins`.

Installing a Plugin

1. Download a binary or the source code for a plugin from a trusted provider.

   Note: The cf CLI requires a binary file compiled from source code written in Go. If you download source code, you must compile the code to create a binary.

2. Run `cf install-plugin BINARY-FILENAME` to install a plugin. Replace `BINARY-FILENAME` with the path to and name of your binary file.

   Note: You cannot install a plugin that has the same name or that uses the same command as an existing plugin. You will be prompted to uninstall the existing plugin.

   Note: The cf CLI prohibits you from implementing any plugin that uses a native cf CLI command name or alias. For example, if you attempt to install a third-party plugin that includes the command `cf push`, the cf CLI halts the installation.

Running a Plugin Command

Use the contents of the `cf help` **CLI plugin management** and **Commands offered by installed plugins** sections to manage plugins and run plugin commands.

1. Run `cf plugins` to list all installed plugins and all commands that the plugins provide.

2. Run `cf PLUGIN-COMMAND` to execute a plugin command.

Checking for Plugin Updates

Run `cf plugins --outdated` to check all registered plugin repositories for newer versions of currently installed plugins.

Example:

```
$ cf plugins --outdated
Searching CF-Community, company-repo for newer versions of installed plugins...
plugin        version   latest version
coffeemaker     1.1.2      1.2.0
Use 'cf install-plugin' to update a plugin to the latest version.
```

For more information about the `cf plugins` command, see cf plugins.

Uninstalling a Plugin

Use the `PLUGIN-NAME` to remove a plugin, not the `BINARY-FILENAME`.

1. Run `cf plugins` to view the names of all installed plugins.

2. Run `cf uninstall-plugin PLUGIN-NAME` to remove a plugin.

Adding a Plugin Repository

Run `cf add-plugin-repo REPOSITORY-NAME-URL` to add a plugin repository.

Example:

```
$ cf add-plugin-repo CF-Community https://plugins.cloudfoundry.org
https://plugins.cloudfoundry.org added as CF-Community
```

Listing Available Plugin Repositories

Run cf list-plugin-repos to view your available plugin repositories.

Example:

```
$ cf list-plugin-repos
OK
Repo Name     Url
CF-Community   https://plugins.cloudfoundry.org
```

Listing All Plugins by Repository

Run cf repo-plugins to show all plugins from all available repositories.

Troubleshooting

The cf CLI provides the following error messages to help you troubleshoot installation and usage issues. Third-party plugins can provide their own error messages.

Permission Denied

If you receive a `permission denied` error message, you lack required permissions to the plugin. You must have `read` and `execute` permissions to the plugin binary file.

Plugin Command Collision

Plugin names and commands must be unique. The CLI displays an error message if you attempt to install a plugin with a non-unique name or command.

If the plugin has the same name or command as a currently installed plugin, you must first uninstall the existing plugin to install the new plugin.

If the plugin has a command with the same name as a native cf CLI command or alias, you cannot install the plugin.

Developing cf CLI Plugins

Page last updated:

Users can create and install Cloud Foundry Command Line Interface (cf CLI) plugins to provide custom commands. These plugins can be submitted and shared to the CF Community repository.

Requirements

Using plugins requires cf CLI v.6.7 or higher. Refer to the Installing the Cloud Foundry Command Line Interface topic for information about downloading, installing, and uninstalling the cf CLI.

Installing the Architecture

1. Implement the predefined plugin interface.
2. Clone the template repository. You will need the basic GO plugin.

Initializing the Plugin

To initialize a plugin, call `plugin.Start(new(MyPluginStruct))` from within the `main()` method of your plugin. The `plugin.Start(...)` function requires a new reference to the `struct` that implements the defined interface.

Invoking cf CLI Commands

Invoke cf CLI commands with `cliConnection.CliCommand([]args)` from within a plugin's `Run(...)` method. `The Run(...)` method receives the `cliConnection` as its first argument. The `cliConnection.CliCommand([]args)` returns the output printed by the command and an error.

The output is returned as a slice of strings. The error will be present if the call to the cf CLI command fails.

For more information, see the API commands documentation.

Installing a Plugin

To install a plugin, run `cf install-plugin PATH_TO_PLUGIN_BINARY`.

For additional information about developing plugins, see the plugin development guide.

Cloud Foundry CLI Reference Guide

Name

cf - A command line tool to interact with Cloud Foundry

Usage

cf [global options] command [arguments...] [command options]

Version

6.33.1+c77e55743.2017-12-15

Getting Started

Command	Description
help	Show help
version	Print the version
login	Log user in
logout	Log user out
passwd	Change user password
target	Set or view the targeted org or space
api	Set or view target api url
auth	Authenticate user non-interactively

Apps

Command	Description
apps	List all apps in the target space
app	Display health and status for an app
push	Push a new app or sync changes to an existing app
scale	Change or view the instance count, disk space limit, and memory limit for an app
delete	Delete an app
rename	Rename an app
start	Start an app
stop	Stop an app
restart	Stop all instances of the app, then start them again. This causes downtime.
restage	Recreate the app's executable artifact using the latest pushed app files and the latest environment (variables, service bindings, buildpack, stack, etc.)
restart-app-instance	Terminate, then restart an app instance
run-task	Run a one-off task on an app
tasks	List tasks of an app

Command	Description
terminate-task	Terminate a running task of an app
events	Show recent app events
files	Print out a list of files in a directory or the contents of a specific file of an app running on the DEA backend
logs	Tail or show recent logs for an app
env	Show all env variables for an app
set-env	Set an env variable for an app
unset-env	Remove an env variable
stacks	List all stacks (a stack is a pre-built file system, including an operating system, that can run apps)
stack	Show information for a stack (a stack is a pre-built file system, including an operating system, that can run apps)
copy-source	Copies the source code of an application to another existing application (and restarts that application)
create-app-manifest	Create an app manifest for an app that has been pushed successfully
get-health-check	Show the type of health check performed on an app
set-health-check	Change type of health check performed on an app
enable-ssh	Enable ssh for the application
disable-ssh	Disable ssh for the application
ssh-enabled	Reports whether SSH is enabled on an application container instance
ssh	SSH to an application container instance

Services

Command	Description
marketplace	List available offerings in the marketplace
services	List all service instances in the target space
service	Show service instance info
create-service	Create a service instance
update-service	Update a service instance
delete-service	Delete a service instance
rename-service	Rename a service instance
create-service-key	Create key for a service instance
service-keys	List keys for a service instance
service-key	Show service key info
delete-service-key	Delete a service key
bind-service	Bind a service instance to an app
unbind-service	Unbind a service instance from an app
bind-route-service	Bind a service instance to an HTTP route
unbind-route-service	Unbind a service instance from an HTTP route
create-user-provided-service	Make a user-provided service instance available to CF apps
update-user-provided-service	Update user-provided service instance

Orgs

Command	Description

Command	Description
orgs	List all orgs
org	Show org info
create-org	Create an org
delete-org	Delete an org
rename-org	Rename an org

Spaces

Command	Description
spaces	List all spaces in an org
space	Show space info
create-space	Create a space
delete-space	Delete a space
rename-space	Rename a space
allow-space-ssh	Allow SSH access for the space
disallow-space-ssh	Disallow SSH access for the space
space-ssh-allowed	Reports whether SSH is allowed in a space

Domains

Command	Description
domains	List domains in the target org
create-domain	Create a domain in an org for later use
delete-domain	Delete a domain
create-shared-domain	Create a domain that can be used by all orgs (admin-only)
delete-shared-domain	Delete a shared domain
router-groups	List router groups

Routes

Command	Description
routes	List all routes in the current space or the current organization
create-route	Create a url route in a space for later use
check-route	Perform a simple check to determine whether a route currently exists or not
map-route	Add a url route to an app
unmap-route	Remove a url route from an app
delete-route	Delete a route
delete-orphaned-routes	Delete all orphaned routes (i.e. those that are not mapped to an app)

Network Policies

Command	Description
network-policies	List direct network traffic policies
	Create policy to allow direct network traffic from one app to another

Command	Description
add-network-policy	
remove-network-policy	Remove network traffic policy of an app

Buildpacks

Command	Description
buildpacks	List all buildpacks
create-buildpack	Create a buildpack
update-buildpack	Update a buildpack
rename-buildpack	Rename a buildpack
delete-buildpack	Delete a buildpack

User Admin

Command	Description
create-user	Create a new user
delete-user	Delete a user
org-users	Show org users by role
set-org-role	Assign an org role to a user
unset-org-role	Remove an org role from a user
space-users	Show space users by role
set-space-role	Assign a space role to a user
unset-space-role	Remove a space role from a user

Org Admin

Command	Description
quotas	List available usage quotas
quota	Show quota info
set-quota	Assign a quota to an org
create-quota	Define a new resource quota
delete-quota	Delete a quota
update-quota	Update an existing resource quota
share-private-domain	Share a private domain with an org
unshare-private-domain	Unshare a private domain with an org

Space Admin

Command	Description
space-quotas	List available space resource quotas
space-quota	Show space quota info
create-space-quota	Define a new space resource quota
update-space-quota	Update an existing space quota
delete-space-quota	Delete a space quota definition and unassign the space quota from all spaces

Pivotal

Command	Description
set-space-quota	Assign a space quota definition to a space
unset-space-quota	Unassign a quota from a space

Service Admin

Command	Description
service-auth-tokens	List service auth tokens
create-service-auth-token	Create a service auth token
update-service-auth-token	Update a service auth token
delete-service-auth-token	Delete a service auth token
service-brokers	List service brokers
create-service-broker	Create a service broker
update-service-broker	Update a service broker
delete-service-broker	Delete a service broker
rename-service-broker	Rename a service broker
migrate-service-instances	Migrate service instances from one service plan to another
purge-service-offering	Recursively remove a service and child objects from Cloud Foundry database without making requests to a service broker
purge-service-instance	Recursively remove a service instance and child objects from Cloud Foundry database without making requests to a service broker
service-access	List service access settings
enable-service-access	Enable access to a service or service plan for one or all orgs
disable-service-access	Disable access to a service or service plan for one or all orgs

Security Group

Command	Description
security-group	Show a single security group
security-groups	List all security groups
create-security-group	Create a security group
update-security-group	Update a security group
delete-security-group	Deletes a security group
bind-security-group	Bind a security group to a particular space, or all existing spaces of an org
unbind-security-group	Unbind a security group from a space
bind-staging-security-group	Bind a security group to the list of security groups to be used for staging applications
staging-security-groups	List security groups in the staging set for applications
unbind-staging-security-group	Unbind a security group from the set of security groups for staging applications
bind-running-security-group	Bind a security group to the list of security groups to be used for running applications
running-security-groups	List security groups in the set of security groups for running applications
unbind-running-security-group	Unbind a security group from the set of security groups for running applications

Environment Variable Groups

Command	Description
running-environment-variable-group	Retrieve the contents of the running environment variable group
staging-environment-variable-group	Retrieve the contents of the staging environment variable group
set-staging-environment-variable-group	Pass parameters as JSON to create a staging environment variable group
set-running-environment-variable-group	Pass parameters as JSON to create a running environment variable group

Isolation Segments

Command	Description
isolation-segments	List all isolation segments
create-isolation-segment	Create an isolation segment
delete-isolation-segment	Delete an isolation segment
enable-org-isolation	Entitle an organization to an isolation segment
disable-org-isolation	Revoke an organization's entitlement to an isolation segment
set-org-default-isolation-segment	Set the default isolation segment used for apps in spaces in an org
reset-org-default-isolation-segment	Reset the default isolation segment used for apps in spaces of an org
set-space-isolation-segment	Assign the isolation segment for a space
reset-space-isolation-segment	Reset the space's isolation segment to the org default

Feature Flags

Command	Description
feature-flags	Retrieve list of feature flags with status
feature-flag	Retrieve an individual feature flag with status
enable-feature-flag	Allow use of a feature
disable-feature-flag	Prevent use of a feature

Advanced

Command	Description
curl	Executes a request to the targeted API endpoint
config	Write default values to the config
oauth-token	Retrieve and display the OAuth token for the current session
ssh-code	Get a one time password for ssh clients

Add/remove Plugin Repository

Command	Description
add-plugin-repo	Add a new plugin repository
remove-plugin-repo	Remove a plugin repository
list-plugin-repos	List all the added plugin repositories
repo-plugins	List all available plugins in specified repository or in all added repositories

Add/remove Plugin

Command	Description
plugins	List commands of installed plugins
install-plugin	Install CLI plugin
uninstall-plugin	Uninstall CLI plugin

Environment Variables

Variable	Description
CF_COLOR=false	Do not colorize output
CF_DIAL_TIMEOUT=5	Max wait time to establish a connection, including name resolution, in seconds
CF_HOME=path/to/dir/	Override path to default config directory
CF_PLUGIN_HOME=path/to/dir/	Override path to default plugin config directory
CF_TRACE=true	Print API request diagnostics to stdout
CF_TRACE=path/to/trace.log	Append API request diagnostics to a log file
https_proxy=proxy.example.com:8080	Enable HTTP proxying for API requests

Global Options

Option	Description
–help, -h	Show help
-v	Print API request diagnostics to stdout

Apps (experimental)

Command	Description
v3-apps	List all apps in the target space
v3-app	Display health and status for an app
v3-create-app	Create a V3 App
v3-push	Push a new app or sync changes to an existing app
v3-scale	Change or view the instance count, disk space limit, and memory limit for an app
v3-delete	Delete a V3 App
v3-start	Start an app
v3-stop	Stop an app
v3-restart	Stop all instances of the app, then start them again. This causes downtime.
v3-stage	Create a new droplet for an app
v3-restart-app-instance	Terminate, then instantiate an app instance
v3-droplets	List droplets of an app
v3-set-droplet	Set the droplet used to run an app
v3-set-env	Set an env variable for an app
v3-unset-env	Remove an env variable from an app
v3-get-health-check	Show the type of health check performed on an app
v3-set-health-check	Change type of health check performed on an app's process

Command	Description
v3-packages	List packages of an app
v3-create-package	Uploads a V3 Package
v2-push	Push a new app or sync changes to an existing app

Developer Guide

This guide has instructions for pushing an application to Cloud Foundry and making the application work with any available cloud-based services it uses, such as databases, email, or message servers. The core of this guide is the Deploy an Application process guide, which provides end-to-end instructions for deploying and running applications on Cloud Foundry, including tips for troubleshooting deployment and application health issues.

Before you can use the instructions in this document, you must have an account on your Cloud Foundry instance.

Preparing Applications for the Cloud

- Considerations for Designing and Running an Application in the Cloud

Check out the 15-minute Getting Started with PCF tutorial for learning Pivotal Cloud Foundry app deployment concepts.

Deploying and Managing Applications

- Deploy an Application
- Deploy a Large Application
- Deploy an App with Docker
- Starting, Restarting, and Restaging Applications
- Application Container Lifecycle
- Routes and Domains
- Changing Stacks
- Deploying with Application Manifests
- Using Application Health Checks
- Scaling an Application Using cf scale
- Running Tasks
- Scaling an Application Using App Autoscaler
- Cloud Foundry Environment Variables
- Using Blue-Green Deployment to Reduce Downtime and Risk
- Application Logging in Cloud Foundry
- Troubleshooting Application Deployment and Health
- Application SSH Overview
- Accessing Apps with SSH
- Accessing Services with SSH
- Trusted System Certificates
- Cloud Controller API Client Libraries
- Using Experimental cf CLI Commands

Services

- Services Overview
- Delivering Service Credentials to an Application
- Managing Service Instances
- Managing Service Keys
- User-Provided Service Instances
- Streaming Application Logs to Log Management Services
- Service-Specific Instructions for Streaming Application Logs
- Streaming Application Logs to Splunk

- Streaming Application Logs with Fluentd
- Streaming Application Logs to Azure OMS Log Analytics
- Configuring Play Framework Service Connections
- Migrating a Database in Cloud Foundry
- Using an External Filesystem (Volume Services)

Considerations for Designing and Running an Application in the Cloud

Page last updated:

Application Design for the Cloud

Applications written in supported application frameworks often run unmodified on Cloud Foundry, if the application design follows a few simple guidelines. Following these guidelines makes an application cloud-friendly, and facilitates deployment to Cloud Foundry and other cloud platforms.

The following guidelines represent best practices for developing modern applications for cloud platforms. For more detailed reading about good app design for the cloud, see The Twelve-Factor App.

For more information about the features of HTTP routing handled by the Cloud Foundry router, see the HTTP Routing topic. For more information about the lifecycle of application containers, see the Application Container Lifecycle topic.

Avoid Writing to the Local File System

Applications running on Cloud Foundry should not write files to the local file system for the following reasons:

- **Local file system storage is short-lived.** When an application instance crashes or stops, the resources assigned to that instance are reclaimed by the platform including any local disk changes made since the app started. When the instance is restarted, the application will start with a new disk image. Although your application can write local files while it is running, the files will disappear after the application restarts.
- **Instances of the same application do not share a local file system.** Each application instance runs in its own isolated container. Thus a file written by one instance is not visible to other instances of the same application. If the files are temporary, this should not be a problem. However, if your application needs the data in the files to persist across application restarts, or the data needs to be shared across all running instances of the application, the local file system should not be used. We recommend using a shared data service like a database or blobstore for this purpose.

For example, instead of using the local file system, you can use a Cloud Foundry service such as the MongoDB document database or a relational database like MySQL or Postgres. Another option is to use cloud storage providers such as Amazon S3, Google Cloud Storage, Dropbox, or Box. If your application needs to communicate across different instances of itself, consider a cache like Redis or a messaging-based architecture with RabbitMQ.

If you must use a file system for your application because for example, your application interacts with other applications through a network attached file system, or because your application is based on legacy code that you cannot rewrite, consider using Volume Services to bind a network attached file system to your application.

Cookies Accessible across Applications

In an environment with shared domains, cookies might be accessible across applications.

Many tracking tools such as Google Analytics and Mixpanel use the highest available domain to set their cookies. For an application using a shared domain such as `example.com`, a cookie set to use the highest domain has a `Domain` attribute of `.example.com` in its HTTP response header. For example, an application at `my-app.shared-domain.example.com` might be able to access the cookies for an application at `your-app.shared-domain.example.com`.

You should decide whether or not you want your applications or tools that use cookies to set and store the cookies at the highest available domain.

Port Considerations

Clients connect to applications running on Cloud Foundry by making requests to URLs associated with the application. Cloud Foundry allows HTTP requests to applications on ports 80 and 443. For more information, see the Routes and Domains topic.

Cloud Foundry also supports WebSocket handshake requests over HTTP containing the `Upgrade` header. The Cloud Foundry router handles the upgrade and initiates a TCP connection to the application to form a WebSocket connection.

To support WebSockets, the operator must configure the load balancer correctly. Depending on the configuration, clients may have to use a different port for WebSocket connections, such as port 4443, or a different domain name. For more information, see the Supporting WebSockets topic.

Cloud Foundry Updates and Your Application

For application management purposes, Cloud Foundry may need to stop and restart your application instances. If this occurs, Cloud Foundry performs the following steps:

1. Cloud Foundry sends a single `termination signal` to the root process that your start command invokes.

2. Cloud Foundry waits 10 seconds to allow your application to cleanly shut down any child processes and handle any open connections.

3. After 10 seconds, Cloud Foundry forcibly shuts down your application.

Your application should accept and handle the termination signal to ensure that it shuts down gracefully.

Ignore Unnecessary Files When Pushing

By default, when you push an application, all files in the application's project directory tree are uploaded to your Cloud Foundry instance, except version control and configuration files or folders with the following names:

- `.cfignore`
- `_darcs`
- `.DS_Store`
- `.git`
- `.gitignore`
- `.hg`
- `manifest.yml`
- `.svn`

In addition to these, if API request diagnostics are directed to a log file and the file is within the project directory tree, it is excluded from the upload. You can direct these API request diagnostics to a log file using `cf config --trace` or the `CF_TRACE` environment variable.

If the application directory contains other files, such as `temp` or `log` files, or complete subdirectories that are not required to build and run your application, you might want to add them to a `.cfignore` file to exclude them from upload. Especially with a large application, uploading unnecessary files can slow application deployment.

To use a `.cfignore` file, create a text file named `.cfignore` in the root of your application directory structure. In this file, specify the files or file types you wish to exclude from upload. For example, these lines in a `.cfignore` file exclude the "tmp" and "log" directories.

```
tmp/
log/
```

The file types you will want to exclude vary, based on the application frameworks you use. For examples of commonly-used `.gitignore` files, see [https://github.com/github/gitignore].

Run Multiple Instances to Increase Availability

When a Diego cell is upgraded, the applications running on it are shut down gracefully, then restarted on another Diego cell. To avoid the risk of an application being unavailable during a Cloud Foundry upgrade processes, you should run more than one instance of the application.

Using Buildpacks

A buildpack consists of bundles of detection and configuration scripts that provide framework and runtime support for your applications. When you deploy an application that needs a buildpack, Cloud Foundry installs the buildpack on the Diego cell where the application runs.

For more information, see the Buildpacks topic.

Deploy an Application

Page last updated:

Note: See the buildpacks documentation for deployment guides specific to your app language or framework, such as the Getting Started Deploying Ruby on Rails Apps guide.

Overview of Deployment Process

You deploy an app to Cloud Foundry by running a `cf push` command from the Cloud Foundry Command Line Interface (cf CLI). Refer to the Installing the cf CLI topic for more information. Between the time that you run `cf push` and the time that the app is available, Cloud Foundry performs the following tasks:

- Uploads and stores app files
- Examines and stores app metadata
- Creates a “droplet” (the Cloud Foundry unit of execution) for the app
- Selects an appropriate Diego cell to run the droplet
- Starts the app

For more information about the lifecycle of an app, see the Application Container Lifecycle topic.

An app that uses services, such as a database, messaging, or email server, is not fully functional until you provision the service and, if required, bind the service to the app. For more information about services, see the Services Overview topic.

Step 1: Prepare to Deploy

Before you deploy your app to Cloud Foundry, make sure that:

- Your app is *cloud-ready*. Cloud Foundry behaviors related to file storage, HTTP sessions, and port usage may require modifications to your app.
- All required app resources are uploaded. For example, you may need to include a database driver.
- Extraneous files and artifacts are excluded from upload. You should explicitly exclude extraneous files that reside within your app directory structure, particularly if your app is large.
- An instance of every service that your app needs has been created.
- Your Cloud Foundry instance supports the type of app you are going to deploy, or you have the URL of an externally available buildpack that can stage the app.

For help preparing to deploy your app, see:

- Considerations for Designing and Running an Application in the Cloud
- Buildpacks

Step 2: Know Your Credentials and Target

Before you can push your app to Cloud Foundry you need to know:

- The API endpoint for your Cloud Foundry instance. Also known as the target URL, this is the URL of the Cloud Controller in your PAS instance.
- Your username and password for your Cloud Foundry instance.
- The organization and space where you want to deploy your app. A Cloud Foundry workspace is organized into organizations, and within them, spaces. As a Cloud Foundry user, you have access to one or more organizations and spaces.

Step 3: (Optional) Configure Domains

Cloud Foundry directs requests to an app using a route, which is a URL made up of a host and a domain.

- The name of an app is the default host for that app, unless you specify the host name with the `-n` flag.
- Every app is deployed to an app space that belongs to a domain. Every Cloud Foundry instance has a default domain defined. You can specify a non-default, or custom, domain when deploying, provided that the domain is registered and is mapped to the organization which contains the target app space.

> **Note**: CF allows app names, but not app URLs, to include underscores. CF converts underscores to hyphens when setting a default app URL from an app name.

- The URL for your app must be unique from other apps hosted by PAS. Use the following options with the cf CLI to help create a unique URL:
  - `-n` to assign a different HOST name for the app
  - `--random-route` to create a URL that includes the app name and random words

> **Note**: Use `cf help push` to view other options for this command.

For more information about domains, see Routes and Domains.

Step 4: Determine Deployment Options

Before you deploy, you need to decide on the following:

- **Name**: You can use any series of alpha-numeric characters as the name of your app.
- **Instances**: Generally speaking, the more instances you run, the less downtime your app will experience. If your app is still in development, running a single instance can simplify troubleshooting. For any production app, we recommend a minimum of two instances.
- **Memory Limit**: The maximum amount of memory that each instance of your app can consume. If an instance exceeds this limit, Cloud Foundry restarts the instance.

  > **Note**: Initially, Cloud Foundry immediately restarts any instances that exceed the memory limit. If an instance repeatedly exceeds the memory limit in a short period of time, Cloud Foundry delays restarting the instance.

- **Start Command**: This is the command that Cloud Foundry uses to start each instance of your app. This start command varies by app framework.
- **Subdomain (host) and Domain**: The route, which is the combination of subdomain and domain, must be globally unique. This is true whether you specify a portion of the route or allow Cloud Foundry to use defaults.
- **Services**: Apps can bind to services such as databases, messaging, and key-value stores. Apps are deployed into app spaces. An app can only bind to a service that has an existing instance in the target app space.

Define Deployment Options

You can define deployment options on the command line, in a manifest file, or both together. See Deploying with Application Manifests to learn how app settings change from push to push, and how command-line options, manifests, and commands like `cf scale` interact.

When you deploy an app while it is running, Cloud Foundry stops all instances of that app and then deploys. Users who try to run the app get a "404 not found" message while `cf push` runs. Stopping all instances is necessary to prevent two versions of your code from running at the same time. A worst-case example would be deploying an update that involved a database schema migration, because instances running the old code would not work and users could lose data.

Cloud Foundry uploads all app files except version control files and folders with names such as `.svn`, `.git`, and `_darcs`. To exclude other files from upload, specify them in a `.cfignore` file in the directory where you run the push command. For more information, see the Ignore Unnecessary Files When Pushing section of the Considerations for Designing and Running an Application in the Cloud topic.

For more information about the manifest file, see the Deploying with Application Manifests topic.

Configure Pre-Runtime Hooks

> **Note**: The Java buildpack does not support pre-runtime hooks.

To configure pre-runtime hooks, create a file named `.profile` and place it in the root of your app directory. If the directory includes a `.profile` script, then Cloud Foundry executes it immediately before each instance of your app starts. Because the `.profile` script executes after the buildpack, the script has access to the language runtime environment created by the buildpack.

> **Note**: Your app root directory may also include a `.profile.d` directory that contains bash scripts that perform initialization tasks for the buildpack. Developers should not edit these scripts unless they are using a custom buildpack.

You can use the `.profile` script to perform app-specific initialization tasks, such as setting custom environment variables. Environment variables are key-value pairs defined at the operating system level. These key-value pairs provide a way to configure the apps running on a system. For example, any app can access the LANG environment variable to determine which language to use for error messages and instructions, collating sequences, and date formats.

To set an environment variable, add the appropriate bash commands to your `.profile` file. See the example below.

```
# Set the default LANG for your apps
export LANG=en_US.UTF-8
```

> **Note**: If you are using a PHP buildpack version prior to v4.3.18, the buildpack does not execute your PHP app's `.profile` script. Your PHP app will host the `.profile` script's contents. This means that any PHP app staged using the affected PHP buildpack versions can leak credentials placed in the `.profile` script.

Step 5: Push the App

Run the following command to deploy an app without a manifest:

`cf push APP-NAME`

If you provide the app name in a manifest, you can reduce the command to `cf push`. See Deploying with Application Manifests.

Because all you have provided is the name of your app, `cf push` sets the number of instances, amount of memory, and other attributes of your app to the default values. You can also use command-line options to specify these and additional attributes.

The following transcript illustrates how Cloud Foundry assigns default values to app when given a `cf push` command.

> **Note**: When deploying your own apps, avoid generic names like `my-app`. Cloud Foundry uses the app name to compose the route to the app, and deployment fails unless the app has a globally unique route.

```
$ cf push my-app
Creating app my-app in org example-org / space development as a.user@shared-domain.example.com...
OK

Creating route my-app.shared-domain.example.com...
OK

Binding my-app.shared-domain.example.com to my-app...
OK

Uploading my-app...
Uploading app: 560.1K, 9 files
OK

Starting app my-app in org example-org / space development as a.user@shared-domain.example.com...
-----> Downloaded app package (552K)
OK
-----> Using Ruby version: ruby-1.9.3
-----> Installing dependencies using Bundler version 1.3.2
       Running: bundle install --without development:test --path
         vendor/bundle --binstubs vendor/bundle/bin --deployment
       Installing rack (1.5.1)
       Installing rack-protection (1.3.2)
       Installing tilt (1.3.3)
       Installing sinatra (1.3.4)
       Using bundler (1.3.2)
       Updating files in vendor/cache
       Your bundle is complete! It was installed into ./vendor/bundle
       Cleaning up the bundler cache.
-----> Uploading droplet (23M)

1 of 1 instances running

App started

Showing health and status for app my-app in org example-org / space development as a.user@shared-domain.example.com...
OK

requested state: started
instances: 1/1
usage: 1G x 1 instances
urls: my-app.shared-domain.example.com

     state     since                    cpu    memory        disk
#0   running   2014-01-24 05:07:18 PM   0.0%   18.5M of 1G   52.5M of 1G
```

Step 6: (Optional) Configure Service Connections

If you bound a service to the app that you deployed, you might need to configure your app with the service URL and credentials. For more information, see the specific documentation for your app framework:

- Ruby
- Node.js
- Spring
- Grails

Step 7: Troubleshoot Deployment Problems

If your app does not start on Cloud Foundry, first ensure that your app can run locally.

You can troubleshoot your app in the cloud using the cf CLI. See Troubleshoot Application Deployment and Health.

Deploying a Large Application

Page last updated:

This topic describes constraints and recommended settings for deploying applications above 750 MB.

Deployment Considerations and Limitations

The deployment process involves uploading, staging, and starting the app. See the Deployment section of the Application Container Lifecycle topic for more information about the default time limits for uploading, staging, and starting an app.

To deploy large apps to PAS, ensure the following:

- The total size of the files to upload for your app does not exceed the maximum app file size that an admin sets in **Ops Manager > PAS > Application Developer Controls**.
- Your network connection speed is sufficient to upload your app within the 15 minute limit. We recommends a minimum speed of 874 KB/s.

> **Note**: PAS provides an authorization token that is valid for a minimum of 20 minutes.

- You allocate enough memory for all instances of your app. Use either the `-m` flag with `cf push` or set an app memory value in your `manifest.yml` file.
- You allocate enough disk space for all instances of your app. Use either the `-k` flag with `cf push` or set a disk space allocation value in your `manifest.yml` file.
- If you use an app manifest file, `manifest.yml`, be sure to specify adequate values for your app for attributes such as app memory, app start timeout, and disk space allocation.
  For more information about using manifests, refer to the Deploying with Application Manifests topic.
- You push only the files that are necessary for your application.
  To meet this requirement, push only the directory for your application, and remove unneeded files or use the `.cfignore` file to specify excluded files.
- You configure Cloud Foundry Command Line Interface (cf CLI) staging, startup, and timeout settings to override settings in the manifest, as necessary.
  - `CF_STAGING_TIMEOUT`: Controls the maximum time that the cf CLI waits for an app to stage after Cloud Foundry successfully uploads and packages the app. Value set in minutes.
  - `CF_STARTUP_TIMEOUT`: Controls the maximum time that the cf CLI waits for an app to start. Value set in minutes.
  - `cf push -t TIMEOUT`: Controls the maximum time that Cloud Foundry allows to elapse between starting an app and the first healthy response from the app. When you use this flag, the cf CLI ignores any app start timeout value set in the manifest. Value set in seconds.

  For more information about using the cf CLI to deploy apps, refer to the Push section of the *Getting Started with the cf CLI* topic.

> **Note**: Changing the timeout setting for the cf CLI does not change the timeout limit for Cloud Foundry server-side jobs such as staging or starting applications. You must change server-side timeouts in the manifest. Because of the differences between the Cloud Foundry and cf CLI timeout values, your app might successfully start even though the cf CLI reports `App failed`. Run `cf apps APP_NAME` to review the actual status of your app.

Default Settings and Limitations Summary Table

This table provides summary information of constraints and default settings to consider when you deploy a large app to PAS.

Setting	Note
App Package Size	Maximum: Set in **Ops Manager > PAS > Application Developer Controls**
Authorization Token Grace Period	Default: 20 minutes, minimum
`CF_STAGING_TIMEOUT`	cf CLI environment variable Default: 15 minutes
`CF_STARTUP_TIMEOUT`	cf CLI environment variable Default: 5 minutes
`cf push -t TIMEOUT`	App start timeout maximum Default: 60 seconds
Disk Space Allocation	Default: 1024 MB
Internet Connection Speed	Recommended Minimum: 874 KB/s

Internet Connection Speed	Recommended Minimum: 874 KB/s

Deploy an App with Docker

Page last updated:

This topic describes how to use the Cloud Foundry Command Line Interface (cf CLI) to push an app with a new or updated Docker image. Cloud Foundry then uses the Docker image to create containers for the app.

See the Using Docker in Cloud Foundry topic for an explanation of how Docker works in Cloud Foundry.

Requirements

To push apps with Docker, you need the following:

- A Cloud Foundry (CF) deployment that has Docker support enabled. To enable Docker support, see the Enable Docker section of *Using Docker in Cloud Foundry*.
- A Docker image that meets the following requirements:
  - The Docker image must contain an `/etc/passwd` file with an entry for the `root` user. In addition, the home directory and the shell for that `root` user must be present in the image filesystem.
  - The total size of the Docker image filesystem layers must not exceed the disk quota for the app. The maximum disk allocation for apps is set by the Cloud Controller. The default maximum disk quota is 2048 MB per app.

  > **Note**: If the total size of the Docker image filesystem layers exceeds the disk quota, the app instances do not start.

- The location of the Docker image on Docker Hub or another Docker registry.
- A registry that supports the Docker Registry HTTP API V2 and presents a valid certificate for HTTPS traffic.

> **Note**: If you want to log in to your app container using the `cf ssh` command, a shell such as `sh` or `bash` must be available in the container. The SSH server in the container looks for the following executables in absolute locations or the `PATH` environment variable: `/bin/bash`, `/usr/local/bin/bash`, `/bin/sh`, `bash`, and `sh`.

Port Configuration

By default, apps listen for connections on the port specified in the PORT environment variable for the app. Cloud Foundry allocates this value dynamically.

When configuring a Docker image for Cloud Foundry, you can control the exposed port and the corresponding value of `PORT` by specifying the EXPOSE directive in the image Dockerfile. If you specify the `EXPOSE` directive, then the corresponding app pushed to Cloud Foundry listens on that exposed port. For example, if you set `EXPOSE` to `7070`, then the app listens for connections on port 7070.

If you do not specify a port in the `EXPOSE` directive, then the app listens on the value of the `PORT` environment variable as determined by Cloud Foundry.

If you set the `PORT` environment variable via an `ENV` directive in a Dockerfile, Cloud Foundry overrides the value with the system-determined value.

Cloud Foundry supports only one exposed port on the image.

Push a Docker Image From Docker Hub

To deploy a Docker image from a Docker Hub repository, run `cf push APP-NAME --docker-image REPO/IMAGE:TAG`. Replace the placeholder values in the command as follows:

- `APP-NAME`: The name of the app being pushed
- `REPO`: The name of the repository where the image is stored
- `IMAGE`: The name of an image from Docker Hub
- `TAG`: (Optional) The tag or version for the image

For example, the following command pushes the `my-image` image from Docker Hub to a Cloud Foundry app:

```
$ cf push my-app --docker-image cloudfoundry/my-image
```

Push a Docker Image from a Private Registry

As an alternative to Docker Hub, you can use any Docker image registry that presents a valid certificate for HTTPS traffic, such as a company-internal Docker registry.

To deploy a Docker image using a specified Docker registry, run `cf push APP-NAME --docker-image MY-PRIVATE-REGISTRY.DOMAIN:PORT/REPO/IMAGE:TAG`.

Replace the placeholder values in the command as follows:

- `APP-NAME`: The name of the app being pushed
- `MY-PRIVATE-REGISTRY.DOMAIN`: The path to the Docker registry
- `PORT`: The port where the registry serves traffic
- `REPO`: The name of the repository where the image is stored
- `IMAGE`: The name of the image being pushed
- `TAG`: (Optional) The tag or version for the image

For example, the following command pushes the `v2` version of the `my-image` image from the `my-repo` repository of the `internal-registry.example.com` registry on port `5000`:

```
$ cf push my-app --docker-image internal-registry.example.com:5000/my-repo/my-image:v2
```

Push a Docker Image From a Registry with Authentication

Many Docker registries control access to Docker images by authenticating with a username and password. Follow the steps below to deploy a Docker image with registry authentication:

1. Make sure the `CF_DOCKER_PASSWORD` environment variable is set to the Docker registry user password.

2. Run the following command:
   `$ CF_DOCKER_PASSWORD=YOUR-PASSWORD cf push APP-NAME --docker-image REPO/IMAGE:TAG --docker-username USER`
   Replace the placeholder values in the command as follows:
   - `YOUR-PASSWORD`: The password to use for authentication with the registry
   - `APP-NAME`: The name of the app being pushed
   - `REPO`: The repository where the image is stored:
     - For Docker Hub, this is just the repository name
     - For a private registry, this includes the registry address and port, as described in Push a Docker Image from a Private Registry, in the format: `MY-PRIVATE-REGISTRY.DOMAIN:PORT/REPO`
   - `IMAGE`: The name of the image being pushed
   - `TAG`: (Optional) The tag or version for the image
   - `USER`: The username to use for authentication with the registry

Push a Docker Image From Google Container Registry (GCR)

PCF supports pushing apps from images hosted on Google Container Registry (GCR) service. This feature requires that you use json_key based authentication.

Step 1: Authenticate with GCR

To authenticate with GCR using PCF, you must create a JSON key file and associate it with your project:

1. Create a GCP service account. See Creating and Enabling Service Accounts for Instances.

```
$ gcloud iam service-accounts create MY-ACCOUNT --display-name "MY DISPLAY NAME"
```

2. Create a JSON key file and associate it with the service account:

```
$ gcloud iam service-accounts keys create key.json --iam-account=MY-ACCOUNT@MY-PROJECT-ID.iam.gserviceaccount.com
```

3. Set your project ID:

```
$ gcloud config set project MY-PROJECT-ID
```

4. Add the IAM policy binding for your project and service account:

```
gcloud projects add-iam-policy-binding MY-PROJECT --member serviceAccount:MY-ACCOUNT@MY-PROJECT-ID.iam.gserviceaccount.com --role roles/storage.objectViewer
```

Step 2: Deploy the GCP Image

Run the following command to deploy your GCR image using the cf CLI:

```
CF_DOCKER_PASSWORD="$(cat key.json)" cf push APP-NAME --docker-image docker://MY-REGISTRY-URL/MY-PROJECT/MY-IMAGE-NAME --docker-username _json_key
```

Replace the placeholder values in the command as follows:

- `APP-NAME` : The name of the app being pushed
- `MY-REGISTRY-URL` : The URL of your registry
- `MY-PROJECT` : The name of your project
- `MY-IMAGE-NAME` : The name of your image

Note: The `key.json` file must point to the file you created in the previous step.

Note: For information about specifying `MY-REGISTRY-URL` , see Pushing and Pulling Images on the Google Cloud documentation.

Docker Volume Support

Diego supports Docker volumes. For more information about enabling volume support, see the Using an External File System (Volume Services) topic. For information about the limitations of NFS volumes, see the NFS Bosh Volume Release.

Starting, Restarting, and Restaging Applications

Page last updated:

This topic describes how to start, restart, and restage applications in Cloud Foundry.

Start Your Application

To start your application, run the following command from your application root directory:

```
$ cf push YOUR-APP
```

For more information about deploying applications, see the Deploy an Application topic.

Cloud Foundry determines the start command for your application from one of the three following sources:

- The `-c` command-line option in the Cloud Foundry Command Line Interface (cf CLI). See the following example:

```
$ cf push YOUR-APP -c "node YOUR-APP.js"
```

- The `command` attribute in the application manifest. See the following example: `command: node YOUR-APP.js`
- The buildpack, which provides a start command appropriate for a particular type of application.

The source that Cloud Foundry uses depends on factors explained below.

How Cloud Foundry Determines its Default Start Command

The first time you deploy an application, `cf push` uses the buildpack start command by default. After that, `cf push` defaults to whatever start command was used for the previous push.

To override these defaults, provide the `-c` option, or the command attribute in the manifest. When you provide start commands both at the command line and in the manifest, `cf push` ignores the command in the manifest.

Forcing Cloud Foundry To Use the Buildpack Start Command

To force Cloud Foundry to use the buildpack start command, specify a start command of `null`.

You can specify a null start command in one of two ways.

- Using the `-c` command-line option in the cf CLI:

```
$ cf push YOUR-APP -c "null"
```

- Using the `command` attribute in the application manifest: `command: null`

This can be helpful after you have deployed while providing a start command at the command line or the manifest. At this point, a command that you provided, rather than the buildpack start command, has become the default start command. In this situation, if you decide to deploy using the buildpack start command, the `null` command makes that easy.

Start Commands When Migrating a Database

Start commands are used in special ways when you migrate a database as part of an application deployment. See the Migrating a Database in Cloud Foundry topic for more information.

Restart Your Application

To restart your application, run the following command:

```
$ cf restart YOUR-APP
```

Restarting your application stops your application and restarts it with the already compiled droplet. A droplet is a tarball that includes:

- stack
- buildpack
- application source code

The Diego cell unpacks, compiles, and runs a droplet on a container.

Restart your application to refresh the application's environment after actions such as binding a new service to the application or setting an environment variable that only the application consumes. However, if your environment variable is consumed by the buildpack in addition to the application, then you must restage the application for the change to take effect.

Restage Your Application

To restage your application, run the following command:

```
$ cf restage YOUR-APP
```

Restaging your application stops your application and restages it, by compiling a new droplet and starting it.

Restage your application if you have changed the environment in a way that affects your staging process, such as setting an environment variable that the buildpack consumes. The staging process has access to environment variables, so the environment can affect the contents of the droplet.

Restaging your application compiles a new droplet from your application without updating your application source. If you must update your application source, re-push your application by following the steps in the section above.

Application Container Lifecycle

Page last updated:

This topic describes the lifecycle of an application container for Cloud Foundry (CF) deployments running on the Diego architecture.

Deployment

The application deployment process involves uploading, staging, and starting the app in a container. Your app must successfully complete each of these phases within certain time limits. The default time limits for the phases are as follows:

- Upload: 15 minutes
- Stage: 15 minutes
- Start: 60 seconds

> **Note**: Your administrator can change these defaults. Check with your administrator for the actual time limits set for app deployment.

Developers can change the time limit for starting apps through an application manifest or on the command line. For more information, see The timeout attribute section of the Deploying with Application Manifests topic and Using Application Health Checks.

Crash Events

If an app instance crashes, CF automatically restarts it by rescheduling the instance on another container three times. After three failed restarts, CF waits thirty seconds before attempting another restart. The wait time doubles each restart until the ninth restart, and remains at that duration until the 200th restart. After the 200th restart, CF stops trying to restart the app instance.

Evacuation

Certain operator actions require restarting VMs with containers hosting app instances. For example, an operator who updates stemcells or installs a new version of CF must restart all the VMs in a deployment. CF automatically relocates the instances on VMs that are shutting down through a process called evacuation. CF recreates the app instances on another VM, waits until they are healthy, and then shuts down the old instances. During an evacuation, developers may see their app instances in a duplicated state for a brief period.

Shutdown

When PCF requests a shutdown of your app instance, either in response to the command `cf scale APPNAME -i NUMBER-OF-INSTANCES` or because of a system event, CF sends the app process in the container a SIGTERM. The process has ten seconds to shut down gracefully. If the process has not exited after ten seconds, CF sends a SIGKILL.

Apps must finish their in-flight jobs within ten seconds of receiving the SIGTERM before CF terminates the app with a SIGKILL. For instance, a web app must finish processing existing requests and stop accepting new requests.

Routes and Domains

Page last updated:

This topic describes how routes and domains work in Pivotal Application Service, and how developers and administrators configure routes and domains for their applications using the Cloud Foundry Command Line Interface (cf CLI).

For more information about routing capabilities in PAS, see the HTTP Routing topic.

Routes

The PAS Gorouter routes requests to applications by associating an app with an address, known as a route. We call this association a **mapping**. Use the cf CLI cf map-route command to associate an app and route.

The routing tier compares each request with a list of all the routes mapped to apps and attempts to find the best match. For example, the Gorouter would make the following matches for the two routes `myapp.shared-domain.example.com` and `myapp.shared-domain.example.com/products`:

Request	Matched Route
http://myapp.shared-domain.example.com	myapp.shared-domain.example.com
http://myapp.shared-domain.example.com/contact	myapp.shared-domain.example.com
http://myapp.shared-domain.example.com/products	myapp.shared-domain.example.com/products
http://myapp.shared-domain.example.com/products/123	myapp.shared-domain.example.com/products
http://products.shared-domain.example.com	No match; 404

The Gorouter does not use a route to match requests until the route is mapped to an app. In the above example, `products.shared-domain.example.com` may have been created as a route in Cloud Foundry, but until it is mapped to an app, requests for the route receive a `404`.

The routing tier knows the location of instances for apps mapped to routes. Once the routing tier determines a route as the best match for a request, it makes a load-balancing calculation using a round-robin algorithm, and forwards the request to an instance of the mapped app.

Developers can map many apps to a single route, resulting in load-balanced requests for the route across all instances of all mapped apps. This approach enables the blue/green zero-downtime deployment strategy. Developers can also map an individual app to multiple routes, enabling access to the app from many URLs.

Routes belong to a space, and developers can only map apps to a route in the same space.

> **Note**: Routes are globally unique. Developers in one space cannot create a route with the same URL as developers in another space, regardless of which orgs control these spaces.

HTTP vs. TCP Routes

> **Note**: By default, PAS only supports routing of HTTP requests to applications.

Routes are considered HTTP if they are created from HTTP domains, and TCP if they are created from TCP domains. See HTTP vs. TCP Shared Domains.

HTTP routes include a domain, an optional hostname, and an optional context path. `shared-domain.example.com`, `myapp.shared-domain.example.com`, and `myapp.shared-domain.example.com/products` are all examples of HTTP routes. Applications should listen to the `localhost` port defined by the `$PORT` environment variable, which is `8080` on Diego. As an example, requests to `myapp.shared-domain.example.com` would be routed to the application container at `localhost:8080`.

- Requests to HTTP routes must be sent to ports 80 or 443.
- Ports cannot be reserved for HTTP routes.

TCP routes include a domain and a route port. A route port is the port clients make requests to. This is not the same port as what an application pushed to Cloud Foundry listens on. `tcp.shared-domain.example.com:60000` is an example of a TCP route. Just as for HTTP routes, applications should listen to the `localhost` port defined by the `$PORT` environment variable, which is `8080` on Diego. As an example, requests to `tcp.shared-domain.example.com:60000` would be routed to the application container at `localhost:8080`.

- Once a port is reserved for a route, it cannot be reserved for another route.
- Hostname and path are not supported for TCP routes.

Create a Route

When a developer creates a route using the cf CLI, PAS determines whether the route is an HTTP or a TCP route based on the domain. To create a HTTP route, a developer must choose an HTTP domain. To create a TCP route, a developer must choose a TCP domain.

Domains in PAS provide a namespace from which to create routes. To list available domains for a targeted organization, use the cf domains command. For more information about domains, see the Domains section.

The following sections describe how developers can create HTTP and TCP routes for different use cases.

Create an HTTP Route with Hostname

In PAS, a hostname is the label that indicates a subdomain of the domain associated with the route. Given a domain `shared-domain.example.com`, a developer can create the route `myapp.shared-domain.example.com` in space `my-space` by specifying the hostname `myapp` with the cf create-route command as shown in this example:

```
$ cf create-route my-space shared-domain.example.com --hostname myapp
Creating route myapp.shared-domain.example.com for org my-org / space my-space as username@example.com...
OK
```

This command instructs PAS to only route requests to apps mapped to this route for the following URLs:

- `http://myapp.shared-domain.example.com`
- `https://myapp.shared-domain.example.com`
- Any path under either of the above URLs, such as `http://myapp.shared-domain.example.com/bar`

Create an HTTP Route without Hostname

This approach creates a route with the same address as the domain itself and is permitted for private domains only. For more information, see the Private Domains section.

A developer can create a route in space `my-space` from the domain `private-domain.example.com` with no hostname with the cf create-route command:

```
$ cf create-route my-space private-domain.example.com
Creating route private-domain.example.com for org my-org / space my-space as username@example.com...
OK
```

If DNS has been configured correctly, this command instructs PAS to route requests to apps mapped to this route from the following URLs:

- `http://private-domain.example.com`
- `https://private-domain.example.com`
- Any path under either of the above URLs, such as `http://private-domain.example.com/foo`

If there are no other routes for the domain, requests to any subdomain, such as `http://foo.private-domain.example.com`, will fail.

A developer can also create routes for subdomains with no hostnames. The following command creates a route in space `my-space` from the subdomain `foo.private-domain.example.com`:

```
$ cf create-route my-space foo.private-domain.example.com
Creating route foo.private-domain.example.com for org my-org / space my-space as username@example.com...
OK
```

Assuming DNS has been configured for this subdomain, this command instructs PAS to route requests to apps mapped to this route from the following URLs:

- `http://foo.private-domain.example.com`
- `https://foo.private-domain.example.com`

- Any path under either of the above URLs, such as `http://foo.private-domain.example.com/foo`

Create an HTTP Route with Wildcard Hostname

An application mapped to a wildcard route acts as a fallback app for route requests if the requested route does not exist. To create a wildcard route, use an asterisk for the hostname.

A developer can create a wildcard route in space `my-space` from the domain `foo.shared-domain.example.com` with the following command:

```
$ cf create-route my-space foo.shared-domain.example.com --hostname '*'
Creating route *.foo.shared-domain.example.com for org my-org / space my-space as username@example.com...
OK
```

If a client sends a request to `http://app.foo.shared-domain.example.com` by accident, attempting to reach `myapp.foo.shared-domain.example.com`, PAS routes the request to the app mapped to the route `*.foo.shared-domain.example.com`.

Create an HTTP Route with a Path

Developers can use paths to route requests for the same hostname and domain to different apps.

A developer can create three routes using the same hostname and domain in the space `my-space` with the following commands:

```
$ cf create-route my-space shared-domain.example.com --hostname store --path products
Creating route store.shared-domain.example.com/products for org my-org / space my-space as username@example.com...
OK

$ cf create-route my-space shared-domain.example.com --hostname store --path orders
Creating route store.shared-domain.example.com/orders for org my-org / space my-space as username@example.com...
OK

$ cf create-route my-space shared-domain.example.com --hostname store
Creating route store.shared-domain.example.com for org my-org / space my-space as username@example.com...
OK
```

The developer can then map the new routes to different apps by following the steps in the Map a Route to Your Application section below.

If the developer maps the first route with path `products` to the `products` app, the second route with path `orders` to the `orders` app, and the last route to the `storefront` app. After this, the following occurs:

- PAS routes requests to `http://store.shared-domain.example.com/products` to the `products` app.
- PAS routes requests to `http://store.shared-domain.example.com/orders` to the `orders` app.
- PAS routes requests to `http://store.shared-domain.example.com` to the `storefront` app.

PAS attempts to match routes with a path first, and then attempts to match host and domain.

> **Note**: Routes with the same domain and hostname but different paths can only be created in the same space. Private domains do not have this limitation.

> **Note**: PAS does not route requests for context paths to the root context of an application. Applications must serve requests on the context path.

Create a TCP Route with a Port

A developer can create a TCP route for `tcp.shared-domain.example.com` on an arbitrary port with the following command. If the clients of the app can accommodate addressing an arbitrary port, then developers should use the `--random-port` to instruct PAS to pick a port for your route.

```
$ cf create-route my-space tcp.shared-domain.example.com --random-port
Creating route tcp.shared-domain.example.com for org my-org / space my-space as user@example.com...
OK
Route tcp.shared-domain.example.com:60034 has been created
```

In this example, PAS routes requests to `tcp.shared-domain.example.com:60034` to apps mapped to this route.

To request a specific port, a developer can use the `--port` flag, so long as the port is not reserved for another space. The following command creates a TCP route for `tcp.shared-domain.example.com` on port 60035:

```
$ cf create-route my-space tcp.shared-domain.example.com --port 60035
Creating route tcp.shared-domain.example.com:60035 for org my-org / space my-space as user@example.com...
OK
```

List Routes

Developers can list routes for the current space with the cf routes command. A route is uniquely identified by the combination of hostname, domain, port, and path.

```
$ cf routes
Getting routes as user@private-domain.example.com ...

space     host    domain                        port  path      type  apps
my-space  myapp   shared-domain.example.com                           myapp
my-space  myapp   private-domain.example.com                          myapp
my-space  store   shared-domain.example.com           /products       products
my-space  store   shared-domain.example.com           /orders         orders
my-space  store   shared-domain.example.com                           storefront
my-space          shared-domain.example.com     60000           tcp   tcp-app
```

Developers can only see routes in spaces where they are members.

Check Routes

Developers cannot create a route that is already taken. To check whether a route is available, developers can use the cf check-route command.

The following command checks whether a route with the hostname `store` and the domain `shared-domain.example.com` and the path `products` exists:

```
$ cf check-route store shared-domain.example.com --path /products
Checking for route...
OK
Route store.shared-domain.example.com/products does exist
```

Map a Route to Your Application

For an app to receive requests to a route, developers must map the route to the app with the cf map-route command. If the route does not already exist, this command creates it.

Developers can create and reserve routes for later use by following the steps in the Manually Map a Route section. Or they can map routes to their app immediately as part of a push by following the steps in the Map a Route with Application Push section.

Note: Changes to route mappings are executed asynchronously. On startup, an application will be accessible at its route within a few seconds. Similarly, upon mapping a new route to a running app, the app will be accessible at this route within a few seconds of the CLI exiting successfully.

Manually Map a Route

Given the following routes and applications:

Route	Application
store.shared-domain.example.com/products	products
store.shared-domain.example.com/orders	orders
store.shared-domain.example.com	storefront
tcp.shared-domain.example.com:60000	tcp-app

The following commands map the above routes to their respective apps. Developers use hostname, domain, and path to uniquely identify a route to map their apps to.

```
$ cf map-route products shared-domain.example.com --hostname store --path products
$ cf map-route orders shared-domain.example.com --hostname store --path orders
$ cf map-route storefront shared-domain.example.com --hostname store
$ cf map-route tcp-app tcp.shared-domain.example.com --port 60000
```

The following command maps the wildcard route `*.foo.shared-domain.example.com` to the app `myfallbackapp`.

```
$ cf map-route myfallbackapp foo.shared-domain.example.com --hostname '*'
```

Map a Route with Application Push

Developers can map a route to their app with the `cf push` command.

If a domain or hostname is not specified, then a route will be created using the app name and the default shared domain (see Shared Domains). The following command pushes the app `myapp`, creating the route `myapp.shared-domain.example.com` from the default shared domain `shared-domain.example.com`. If the route has not already been created in another space this command also maps it to the app.

```
$ cf push myapp
```

To customize the route during `push`, specify the domain using the `-d` flag and the hostname with the `--hostname` flag. The following command creates the `foo.private-domain.example.com` route for `myapp`:

```
$ cf push myapp -d private-domain.example.com --hostname foo
```

To map a TCP route during `push`, specify a TCP domain and request a random port using `--random-route`. To specify a port, push the app without a route, then create and map the route manually by following the steps in the Create a TCP Route with a Port section.

```
$ cf push tcp-app -d tcp.shared-domain.example.com --random-route
```

Map a Route Using Application Manifest

Developers can map a route to their app with a manifest by editing the `route` attribute to specify the host, domain, port and/or path components of the route. For more information, see the Deploying with Application Manifests topic.

Map a Route to Multiple Apps

PAS allows multiple apps, or versions of the same app, to be mapped to the same route. This feature enables Blue-Green deployment. For more information see Using Blue-Green Deployment to Reduce Downtime and Risk.

Routing multiple apps to the same route may cause undesirable behavior in some situations by routing incoming requests randomly to one of the apps on the shared route.

See the Routing Conflict section of the Troubleshooting Application Deployment and Health topic for more information about troubleshooting this problem.

Unmap a Route

Developers can remove a route from an app using the `cf unmap-route` command. The route remains reserved for later use in the space where it was created until the route is deleted.

To unmap an HTTP route from an app, identify the route using the hostname, domain, and path:

```
$ cf unmap-route tcp-app private-domain.example.com --hostname myapp --path mypath
```

To unmap a TCP route from an app, identify the route using the domain and port:

```
$ cf unmap-route tcp-app tcp.shared-domain.example.com --port 60000
```

Delete a Route

Developers can delete a route from a space using the `cf delete-route` command.

To delete a HTTP route, identify the route using the hostname, domain, and path:

```
$ cf delete-route private-domain.example.com --hostname myapp --path mypath
```

To delete a TCP route, identify the route using the domain and port.

```
$ cf delete-route tcp.private-domain.example.com --port 60000
```

Routing Requests to a Specific App Instance

Note: Usage of `X-CF-APP-INSTANCE` is supported only for Diego.

Users can route HTTP requests to a specific application instance using the header `X-CF-APP-INSTANCE`. The format of the header should be `X-CF-APP-INSTANCE: APP_GUID:APP_INDEX`.

`APP_GUID` is an internal identifier for your application. Use the `cf APP-NAME --guid` command to discover the `APP_GUID` for your application.

```
$ cf app myapp --guid
```

`APP_INDEX`, for example `0`, `1`, `2`, or `3`, is an identifier for a particular app instance. Use the CLI command `cf app APP-NAME` to get statistics on each instance of a particular app.

```
$ cf app myapp
```

The following example shows a request made to instance `9` of an application with GUID `5cdc7595-2e9b-4f62-8d5a-a86b92f2df0e` and mapped to route myapp.private-domain.example.com.

```
$ curl myapp.private-domain.example.com -H "X-Cf-App-Instance: 5cdc7595-2e9b-4f62-8d5a-a86b92f2df0e:9"
```

If the cf CLI cannot find the instance the format is incorrect, a `404` status code is returned.

Domains

Note: The term domain in this topic differs from its common use and is specific to Cloud Foundry. Likewise, shared domain and private domain refer to resources with specific meaning in Cloud Foundry. The use of domain name, root domain, and subdomain refers to DNS records.

Domains indicate to a developer that requests for any route created from the domain will be routed to PAS. This requires DNS to be configured out-of-band to resolve the domain name to the IP address of a load balancer configured to forward requests to the CF routers. For more information about configuring DNS, see the DNS for Domains section.

List Domains for an Org

When creating a route, developers will select from domains available to them. Use the `cf domains` command to view a list of available domains for the targeted org:

```
$ cf domains
Getting domains in org my-org as user@example.com... OK
name                    status   type
shared-domain.example.com       shared
tcp.shared-domain.example.com    shared   tcp
private-domain.example.com       owned
```

This example displays three available domains: a shared HTTP domain `shared-domain.example.com`, a shared TCP domain `tcp.shared-domain.example.com`, and a private domain `private-domain.example.com`. See Shared Domains and Private Domains.

HTTP vs. TCP Domains

HTTP domains indicate to a developer that only requests using the HTTP protocol will be routed to applications mapped to routes created from the domain. Routing for HTTP domains is layer 7 and offers features like custom hostnames, sticky sessions, and TLS termination.

TCP domains indicate to a developer that requests over any TCP protocol, including HTTP, will be routed to applications mapped to routes created from the domain. Routing for TCP domains is layer 4 and protocol agnostic, so many features available to HTTP routing are not available for TCP routing. TCP domains are defined as being associated with the TCP Router Group. The TCP Router Group defines the range of ports available to be reserved with TCP Routes. Currently, only Shared Domains can be TCP.

Note: By default, PAS only supports routing of HTTP requests to applications.

Shared Domains

Admins manage shared domains, which are available to users in all orgs of a PAS deployment. An admin can offer multiple shared domains to users. For example, an admin may offer developers the choice of creating routes for their apps from `shared-domain.example.com` and `cf.some-company.com`.

There is not technically a default shared domain. If a developer pushes an app without specifying a domain (see Map a Route with Application Push), a route will be created for it from the first shared domain created in the system. All other operations involving route require the domain be specified (see Routes).

Shared domains are HTTP by default, but can be configured to be TCP when associated with the TCP Router Group.

Create a Shared Domain

Admins can create an HTTP shared domain with the `cf create-shared-domain` command:

```
$ cf create-shared-domain shared-domain.example.com
```

To create a TCP shared domain, first discover the name of the TCP Router Group.

```
$ cf router-groups
Getting router groups as admin ...
name          type
default-tcp   tcp
```

Then create the shared domain using the `--router-group` option to associate the domain with the TCP router group.

```
$ cf create-shared-domain tcp.shared-domain.example.com --router-group default-tcp
```

Delete a Shared Domain

Admins can delete a shared domain from PAS with the `cf delete-shared-domain` command:

```
$ cf delete-shared-domain example.com
```

Private Domains

Org Managers can add private domains, or custom domains, and give members of the org permission to create routes for privately registered domain names. Private domains can be shared with other orgs, enabling users of those orgs to create routes from the domain.

Private domains can be HTTP or HTTPS only. TCP Routing is supported for Shared Domains only.

Create a Private Domain

Org Managers can create a private domain with the following command:

```
$ cf create-domain my-org private-domain.example.com
```

Org Managers can create a private domain for a subdomain with the following command:

```
$ cf create-domain my-org foo.private-domain.example.com
```

Sharing a Private Domain with One or More Orgs

Org Managers can grant or revoke access to a private domain to other orgs if they have permissions for these orgs with the following commands:

```
$ cf share-private-domain test-org private-domain.example.com

$ cf unshare-private-domain test-org private-domain.example.com
```

Delete a Private Domain

Org Managers can delete a domain from PAS with the `cf delete-domain` command:

```
$ cf delete-domain private-domain.example.com
```

Requirements for Parent and Child Domains

In the domain `myapp.shared-domain.example.com` , `shared-domain.example.com` is the parent domain of subdomain `myapp` . Note the following requirements for domains:

- You can only create a private domain that is parent to a private subdomain.
- You can create a shared domain that is parent to either a shared or a private subdomain.

The domain `foo.myapp.shared-domain.example.com` is the child subdomain of `myapp.shared-domain.example.com` . Note the following requirements for subdomains:

- You can create a private subdomain for a private parent domain only if the domains belong to the same org.
- You can create a private subdomain for a shared parent domain.
- You can only create a shared subdomain for a shared parent domain.
- You cannot create a shared subdomain for a private parent domain.

DNS for Domains

To create customized access to your apps, you can map specific or wildcard custom domains to PAS by using your DNS provider.

Mapping Domains to Your Custom Domain

To associate a registered domain name with a domain on PAS, configure a CNAME record with your DNS provider, pointing at any shared domain offered in PAS.

Mapping a Single Domain to Your Custom Domain

To map a single domain to a custom domain to PAS, configure a CNAME record with your DNS provider.

The following table provides some example CNAME record mappings.

Record Set in Custom Domain	Type	Target in PAS
myapp.yourcustomdomain.com.	CNAME	myapp.shared-domain.example.com
www.yourcustomdomain.com.	CNAME	myapp.shared-domain.example.com

After you create the CNAME mapping, your DNS provider routes your custom domain to `myapp.shared-domain.example.com`.

Note: Refer to your DNS provider documentation to determine whether the trailing `.` is required.

Mapping Multiple Subdomains to Your Custom Domain

Use a wildcard CNAME record to point all of the subdomains in your custom domain to shared-domain.example.com.

Each separately configured subdomain has priority over the wildcard configuration.

The following table provides some example wildcard CNAME record mappings.

Record Set in Custom Domain	Type	Target in PAS
*.yourcustomdomain.com.	CNAME	*.shared-domain.example.com
*.yourcustomdomain.com.	CNAME	*.myapp.shared-domain.example.com

If you use a wildcard as the subdomain name, then your DNS provider can route from `*.YOURCUSTOMDOMAIN` to any of the following:

- `*.shared-domain.example.com`
- `foo.myapp.shared-domain.example.com`
- `bar.foo.myapp.shared-domain.example.com`

Configuring DNS for Your Registered Root Domain

To use your root domain (for example, `example.com`) for apps on PAS you can either use custom DNS record types like ALIAS and ANAME, if your DNS provider offers them, or subdomain redirection.

Note: Root domains are also called zone apex domains.

If your DNS provider supports using an ALIAS or ANAME record, configure your root domain with your DNS provider to point at a shared domain in PAS.

Record	Name	Target	Note
ALIAS or ANAME	empty or @	private-domain.example.com.	Refer to your DNS provider documentation to determine whether to use an empty or @ value for the Name entry.

If your DNS provider does not support ANAME or ALIAS records you can use subdomain redirection, also known as domain forwarding, to redirect requests for your root domain to a subdomain configured as a CNAME.

Note: If you use domain forwarding, SSL requests to the root domain may fail if the SSL certificate only matches the subdomain.

Configure the root domain to point at a subdomain such as `www`, and configure the subdomain as a CNAME record pointing at a shared domain in PAS.

Record	Name	Target	Note
URL or Forward	private-domain.example.com	www.private-domain.example.com	This method results in a `301 permanent redirect` to the subdomain you configure.

CNAME	www	myapp.shared-domain.example.com	

Changing Stacks

Page last updated:

A stack is a prebuilt root filesystem (rootfs) that supports a specific operating system. For example, Linux-based systems need `/usr` and `/bin` directories at their root. The stack works in tandem with a buildpack to support applications running in compartments. Under Diego architecture, cell VMs can support multiple stacks.

> **Note:** Docker apps do not use stacks.

Available Stacks

The Linux `cflinuxfs2` stack is derived from Ubuntu Trusty 14.04. Refer to the Github stacks page for supported libraries.

Restaging Applications on a New Stack

For security, stacks receive regular updates to address Common Vulnerabilities and Exposures (CVEs). Apps pick up on these stack changes through new releases of PAS. However, if your app links statically to a library provided in the rootfs, you may have to manually restage it to pick up the changes.

It can be difficult to know what libraries an app statically links to, and it depends on the languages you are using. One example is an app that uses a Ruby or Python binary, and links out to part of the C standard library. If the C library requires an update, you may need to recompile the app and restage it as follows:

1. Use the cf stacks command to list the stacks available in a deployment.

```
$ cf stacks
Getting stacks in org MY-ORG / space development as developer@example.com...
OK

name         description
cflinuxfs2   Cloud Foundry Linux-based filesystem
```

2. To change your stack and restage your application, use the cf push command. For example, to restage your app on the default stack `cflinuxfs2` you can run `cf push MY-APP`:

```
$ cf push MY-APP
Using stack cflinuxfs2...
OK
Creating app MY-APP in org MY-ORG / space development as developer@example.com...
OK
...
requested state: started
instances: 1/1
usage: 1G x 1 instances
urls: MY-APP.cfapps.io
last uploaded: Wed Apr 8 23:40:57 UTC 2015
     state     since                    cpu    memory        disk
#0   running   2015-04-08 04:41:54 PM   0.0%   57.3M of 1G   128.8M of 1G
```

To specify a different stack, append `-s STACKNAME` to the command.

Stacks API

For API information, review the Stacks section of the Cloud Foundry API Documentation.

Deploying with Application Manifests

Page last updated:

You use the Cloud Foundry Command Line Interface tool (cf CLI) `cf push` command to deploy apps. `cf push` deploys apps with a default number of instances, disk space limit, and memory limit. You can override the default values by invoking `cf push` with flags and values, or by specifying key-value pairs in a manifest file.

Manifests provide consistency and reproducibility, and can help you automate deploying apps.

How cf push Finds the Manifest

By default, the `cf push` command deploys an application using a `manifest.yml` file in the current working directory.

For example:

```
$ cf push
Using manifest file /path-to-current-working-directory/manifest.yml
```

If you have created your manifest in a different location, use `cf push -f PATH-TO-MANIFEST`. Replace `PATH-TO-MANIFEST` with the path to your manifest. You can include the name of your mainfest file in `PATH-TO-MANIFEST`.

For example:

```
$ cf push -f ./different-directory-1/different-directory-2/alternate_manifest.yml
Using manifest file ./different-directory-1/different-directory-2/alternate_manifest.yml
```

If you provide a path with no filename, the `cf push` command requires a file named `manifest.yml`.

For example:

```
$ cf push -f ./different-directory-1/different-directory-2/
Using manifest file ./different-directory-1/different-directory-2/manifest.yml
```

Example Manifest

Although you can deploy apps without a manifest, manifests provide consistency and reproducibility. This can be useful when when you want your apps to be portable between different clouds.

Manifests are written in YAML. The manifest below illustrates some YAML conventions, as follows:

- The manifest begins with three dashes.
- The `applications` block begins with a heading followed by a colon.
- The application `name` is preceded by a single dash and one space.
- Subsequent lines in the block are indented two spaces to align with `name`.

```
---
applications:
- name: my-app
  memory: 512M
  instances: 2
```

A minimal manifest requires only an application `name`. To create a valid minimal manifest, remove the `memory` and `instances` properties from this example.

Always Provide an Application Name to cf push

`cf push` requires an application name, which you provide either in a manifest or at the command line.

As described in How cf push Finds the Manifest above, the command `cf push` locates the `manifest.yml` in the current working directory by default, or in the path provided by the `-f` option.

If you do not use a manifest, the minimal `cf push` command looks like this:

```
$ cf push my-app
```

Note: When you provide an application name at the command line, `cf push` uses that application name whether or not there is a different application name in the manifest. If the manifest describes multiple applications, you can push a single application by providing its name at the command line; the cf CLI does not push the others. Use these behaviors for testing.

How cf push Finds the Application

By default, `cf push` recursively pushes the contents of the current working directory. Alternatively, you can provide a path using either a manifest or a command line option.

- If the path is to a directory, `cf push` recursively pushes the contents of that directory instead of the current working directory.
- If the path is to a file, `cf push` pushes only that file.

Note: If you want to push more than a single file, but not the entire contents of a directory, consider using a `.cfignore` file to tell `cf push` what to exclude.

Precedence Between Manifests, Command Line Options, and Most Recent Values

When you push an application for the first time, Cloud Foundry applies default values to any attributes that you do not set in a manifest or `cf push` command line options.

- For example, `cf push my-app` with no manifest might deploy one instance of the app with one gigabyte of memory. In this case the default values for instances and memory are “1” and “1G”, respectively.

Between one push and another, attribute values can change in other ways.

- For example, the `cf scale` command changes the number of instances.

The attribute values on the server at any one time represent the cumulative result of all settings applied up to that point: defaults, attributes in the manifest, `cf push` command line options, and commands like `cf scale`. There is no special name for this resulting set of values on the server. You can think of them as the most recent values.

`cf push` follows rules of precedence when setting attribute values:

- Manifests override most recent values, including defaults.
- Command line options override manifests.

In general, you can think of manifests as just another input to `cf push`, to be combined with command line options and most recent values.

Optional Attributes

This section explains how to describe optional application attributes in manifests. Each of these attributes can also be specified by a command line option. Command line options override the manifest.

buildpack

If your application requires a custom buildpack, you can use the `buildpack` attribute to specify it in one of three ways:

- By name: `MY-BUILDPACK`.
- By GitHub URL: `https://github.com/cloudfoundry/java-buildpack.git`.
- By GitHub URL with a branch or tag: `https://github.com/cloudfoundry/java-buildpack.git#v3.3.0` for the `v3.3.0` tag.

```
---
  ...
  buildpack: buildpack_URL
```

Note: The `cf buildpacks` command lists the buildpacks that you can refer to by name in a manifest or a command line option.

The command line option that overrides this attribute is `-b`.

command

Some languages and frameworks require that you provide a custom command to start an application. Refer to the buildpack documentation to determine if you need to provide a custom start command.

You can provide the custom start command in your application manifest or on the command line. See Starting, Restarting, and Restaging Applications for more information about how Cloud Foundry determines its default start command.

To specify the custom start command in your application manifest, add it in the `command: START-COMMAND` format as the following example shows:

```
---
  ...
  command: bundle exec rake VERBOSE=true
```

The start command you specify becomes the default for your application. To return to using the original default start command set by your buildpack, you must explicitly set the attribute to `null` as follows:

```
---
  ...
  command: null
```

On the command line, use the `-c` option to specify the custom start command as the following example shows:

```
$ cf push my-app -c "bundle exec rake VERBOSE=true"
```

Note: The `-c` option with a value of 'null' forces `cf push` to use the buildpack start command. See Forcing cf push to use the Buildpack Start Command for more information.

If you override the start command for a Buildpack application, Linux uses `bash -c YOUR-COMMAND` to invoke your application. If you override the start command for a Docker application, Linux uses `sh -c YOUR-COMMAND` to invoke your application. Because of this, if you override a start command, you should prefix `exec` to the final command in your custom composite start command.

An app needs to catch termination signals and clean itself up appropriately. Because of the way that shells manage process trees, the use of custom composite shell commands, particularly those that create child processes using `&`, `&&`, `||`, etc., can prevent your app from receiving signals that are sent to the top level bash process.

To resolve this issue, you can use `exec` to replace the bash process with your own process. For example:

- `bin/rake cf:on_first_instance db:migrate && bin/rails server -p $PORT -e $RAILS_ENV` The process tree is bash -> ruby, so on graceful shutdown only the bash process receives the TERM signal, not the ruby process.
- `bin/rake cf:on_first_instance db:migrate && exec bin/rails server -p $PORT -e $RAILS_ENV` Because of the `exec` prefix included on the final command, the ruby process invoked by rails takes over the bash process managing the execution of the composite command. The process tree is only ruby, so the ruby web server receives the TERM signal and can shutdown gracefully for 10 seconds.

In more complex situations, like making a custom buildpack, you may want to use bash `trap`, `wait`, and backgrounded processes to manage your process tree and shut down apps gracefully. In most situations, however, a well-placed `exec` should be sufficient.

disk_quota

Use the `disk_quota` attribute to allocate the disk space for your app instance. This attribute requires a unit of measurement: `M`, `MB`, `G`, or `GB`, in upper case or lower case.

```
---
  ...
  disk_quota: 1024M
```

The command line option that overrides this attribute is `-k`.

docker

If your application is contained in a Docker image, then you may use the `docker` attribute to specify it and an optional Docker username.

This attribute is a combination of `push` options that include `--docker-image` and `--docker-username`.

```
---
  ...
  docker:
    image: docker-image-repository/docker-image-name
    username: docker-user-name
```

The command line option `--docker-image` or `-o` overrides `docker.image`. The command line option `--docker-username` overrides `docker.username`.

The manifest attribute `docker.username` is optional. If it is used, then the password must be provided in the environment variable `CF_DOCKER_PASSWORD`. Additionally, if a Docker username is specified, then a Docker image must also be specified.

Manifest Attributes

The `docker` attribute cannot be used in conjunction with the following attributes: `buildpack` and `path`. An error will result.

domain

Every `cf push` deploys applications to one particular Cloud Foundry instance. Every Cloud Foundry instance may have a shared domain set by an admin. Unless you specify a domain, Cloud Foundry incorporates that shared domain in the route to your application.

You can use the `domain` attribute when you want your application to be served from a domain other than the default shared domain.

```
---
  ...
  domain: unique-example.com
```

The command line option that overrides this attribute is `-d`.

domains

Use the `domains` attribute to provide multiple domains. If you define both `domain` and `domains` attributes, Cloud Foundry creates routes for domains defined in both of these fields.

```
---
  ...
  domains:
  - domain-example1.com
  - domain-example2.org
```

The command line option that overrides this attribute is `-d`.

health-check-http-endpoint

Use the `health-check-http-endpoint` attribute to customize the endpoint for the `http` health check type. If you do not provide a `health-check-http-endpoint` attribute, it uses endpoint '/'.

```
---
  ...
  health-check-type: http
  health-check-http-endpoint: /health
```

health-check-type

Use the `health-check-type` attribute to set the `health_check_type` flag to either `port`, `process` or `http`. If you do not provide a `health-check-type` attribute, it defaults to `port`.

```
---
  ...
  health-check-type: port
```

The command line option that overrides this attribute is `-u`.

The value of `none` is deprecated in favor of `process`.

host

Use the `host` attribute to provide a hostname, or subdomain, in the form of a string. This segment of a route helps to ensure that the route is unique. If you do not provide a hostname, the URL for the app takes the form of `APP-NAME.DOMAIN`.

```
---
  ...
  host: my-app
```

The command line option that overrides this attribute is `-n`.

Note: To specify a wildcard hostname, place quotes around the asterisk. Example: `host: '*'`.

hosts

Use the `hosts` attribute to provide multiple hostnames, or subdomains. Each hostname generates a unique route for the app. You can use `hosts` conjunction with `host`. If you define both attributes, Cloud Foundry creates routes for hostnames defined in both `host` and `hosts`.

```
---
  ...
  hosts:
  - app-host1
  - app-host2
```

The command line option that overrides this attribute is `-n`.

instances

Use the `instances` attribute to specify the number of app instances that you want to start upon push:

```
---
  ...
  instances: 2
```

We recommend that you run at least two instances of any apps for which fault tolerance matters.

The command line option that overrides this attribute is `-i`.

memory

Use the `memory` attribute to specify the memory limit for all instances of an app. This attribute requires a unit of measurement: `M`, `MB`, `G`, or `GB`, in upper case or lower case. For example:

```
---
  ...
  memory: 1024M
```

The default memory limit is 1G. You might want to specify a smaller limit to conserve quota space if you know that your app instances do not require 1G of memory.

The command line option that overrides this attribute is `-m`.

no-hostname

By default, if you do not provide a hostname, the URL for the app takes the form of `APP-NAME.DOMAIN`. If you want to override this and map the root domain to this app then you can set no-hostname as true.

```
---
  ...
  no-hostname: true
```

The command line option that overrides this attribute is `--no-hostname`.

no-route

By default, `cf push` assigns a route to every app. But, some apps process data while running in the background and should not be assigned routes.

You can use the `no-route` attribute with a value of `true` to prevent a route from being created for your app.

```
---
  ...
  no-route: true
```

The command line option that overrides this attribute is `--no-route`.

In the Diego architecture, `no-route` skips creating and binding a route for the app, but does not specify which type of health check to perform. If your app does not listen on a port because it is a worker or a scheduler app, then it does not satisfy the port-based health check and Cloud Foundry marks it as crashed. To prevent this, disable the port-based health check with `cf set-health-check APP_NAME process`.

To remove a route from an existing app, perform the following steps:

1. Remove the route using the `cf unmap-route` command.
2. Push the app again with the `no-route: true` attribute in the manifest or the `--no-route` command line option.

For more information, see Describing Multiple Applications with One Manifest below.

path

You can use the `path` attribute to tell Cloud Foundry the directory location where it can find your application.

The directory specified as the `path`, either as an attribute or as a parameter on the command line, becomes the location where the buildpack `Detect` script executes.

The command line option that overrides this attribute is `-p`.

```
---
  ...
  path: /path/to/application/bits
```

For more information, see the How cf push Finds the Application topic.

random-route

If you push your app without specifying any route-related CLI options or app manifest flags, the cf CLI attempts to generate a route based on the app name, which can cause collisions.

You can use the `random-route` attribute to generate a unique route and avoid name collisions.
When you use `random-route`, the cf CLI generates an HTTP route with a random host (if `host` is not set) or a TCP route with an unused port number.

See the following example use cases:

- You deploy the same app to multiple spaces for testing purposes. In this situation, you can use `random-route` to randomize routes declared with the route attribute in the app manifest.
- You use an app manifest for a classroom training exercise in which multiple users deploy the same app to the same space.

The command line option that overrides this attribute is `--random-route`.

```
---
  ...
  random-route: true
```

routes

Use the `routes` attribute to provide multiple HTTP and TCP routes. Each route for this app is created if it does not already exist.

This attribute is a combination of `push` options that include `--hostname`, `-d`, and `--route-path`.

```
---
  ...
  routes:
  - route: example.com
  - route: www.example.com/foo
  - route: tcp-example.com:1234
```

Manifest Attributes

The `routes` attribute cannot be used in conjunction with the following attributes: `host`, `hosts`, `domain`, `domains`, and `no-hostname`. An error will result.

Push Flag Options

This attribute has unique interactions with different command line options.

Push Flag Option	Resulting Behaviour
`--no-route`	All declared routes are ignored.
`-d`	Overrides DOMAIN part of all declared HTTP and TCP routes.
`--hostname, -n`	Sets or overrides HOSTNAME in all HTTP routes. It has no impact on TCP routes.
`--route-path`	Sets or overrides the PATH in all HTTP routes. It has no impact on TCP routes.
`--random-route`	Sets or overrides the HOSTNAME in all HTTP routes. Sets or overrides the PORT in all TCP routes. The PORT and HOSTNAME will be randomly generated.

stack

Use the `stack` attribute to specify which stack to deploy your application to.

To see a list of available stacks, run `cf stacks` from the cf CLI.

```
---
  ...
  stack: cflinuxfs2
```

The command line option that overrides this attribute is `-s`.

timeout

The `timeout` attribute defines the number of seconds that Cloud Foundry allocates for starting your application. It is related to the `health-check-type` attribute.

For example:

```
---
  ...
  timeout: 80
```

You can increase the timeout length for very large apps that require more time to start. The `timeout` attribute defaults to 60, but you can set it to any value up to the Cloud Controller's `maximum_health_check_timeout` property.

`maximum_health_check_timeout` defaults to 180, but your Cloud Foundry operator can set to any value.

The command line option that overrides the timeout attribute is `-t`.

Environment Variables

The `env` block consists of a heading, then one or more environment variable/value pairs.

For example:

```
---
  ...
  env:
    RAILS_ENV: production
    RACK_ENV: production
```

`cf push` deploys the application to a container on the server. The variables belong to the container environment.

While the application is running, you can modify environment variables.

- View all variables: `cf env my-app`
- Set an individual variable: `cf set-env my-app my-variable_name my-variable_value`
- Unset an individual variable: `cf unset-env my-app my-variable_name my-variable_value`

Environment variables interact with manifests in the following ways:

- When you deploy an application for the first time, Cloud Foundry reads the variables described in the environment block of the manifest and adds them to the environment of the container where the application is staged, and the environment of the container where the application is deployed.
- When you stop and then restart an application, its environment variables persist.

Services

Applications can bind to services such as databases, messaging, and key-value stores.

Applications are deployed into App Spaces. An application can only bind to services instances that exist in the target App Space before the application is deployed.

The `services` block consists of a heading, then one or more service instance names.

Whoever creates the service chooses the service instance names. These names can convey logical information, as in `backend_queue`, describe the nature of the service, as in `mysql_5.x`, or do neither, as in the example below.

```
---
  ...
  services:
   - instance_ABC
   - instance_XYZ
```

Binding to a service instance is a special case of setting an environment variable, namely `VCAP_SERVICES`. See the Bind a Service section of the *Delivering Service Credentials to an Application* topic.

Describing Multiple Applications with One Manifest

You can deploy multiple applications with one `cf push` command by describing the apps in a single manifest.

Suppose you want to deploy two applications called respectively spark and flame, and you want Cloud Foundry to create and start spark before flame. You accomplish this by listing spark first in the manifest.

In this situation there are two sets of bits that you want to push. Let's say that they are `spark.rb` in the spark directory and `flame.rb` in the flame directory. One level up, the `fireplace` directory contains the spark and the flame directories along with the `manifest.yml` file. Your plan is to run the cf CLI from the `fireplace` directory, where you know it can find the manifest.

Now that you have changed the directory structure and manifest location, `cf push` can no longer find your applications by its default behavior of looking in the current working directory. How can you ensure that `cf push` finds the bits you want to push?

The answer is to add a path line to each application description to lead `cf push` to the correct bits. Assume that `cf push` is run from the `fireplace` directory.

For `spark`:

```
---
  ...
  path: ./spark/
```

For `flame`:

```
---
  ...
  path: ./flame/
```

The manifest now consists of two applications blocks.

```
---
# this manifest deploys two applications
# apps are in flame and spark directories
# flame and spark are in fireplace
# cf push should be run from fireplace
applications:
- name: spark
  memory: 1G
  instances: 2
  host: flint-99
  domain: shared-domain.example.com
  path: ./spark/
  services:
  - mysql-flint-99
- name: flame
  memory: 1G
  instances: 2
  host: burnin-77
  domain: shared-domain.example.com
  path: ./flame/
  services:
  - redis-burnin-77
```

Follow these general rules when using a multiple-application manifest:

- Name and completely describe your applications in the manifest.
- Use a `no-route` line in the description of any application that provides background services to another application.
- Do not provide an application name with `cf push`.
- Do not use any command line options with `cf push`.

There are only two narrow exceptions:

- If your manifest is not named `manifest.yml` or not in the current working directory, use the `-f` command line option.
- If you want to push a single application rather than all of the applications described in the manifest, provide the desired application name by running `cf push my-app`.

Minimizing Duplication

In manifests where multiple applications share settings or services, you begin to see content duplicated. While the manifests still work, duplication increases the risk of typographical errors which cause deployment to fail.

The cure for this problem is to “promote” the duplicate content—that is, to move it up above the applications block, where it need appear only once. The promoted content applies to all applications described in the manifest. Note that content *in* the applications block overrides content *above* the applications block, if the two conflict.

The manifest becomes shorter, more readable, and more maintainable.

Notice how much content in the manifest below has been promoted in this way.

```
---
  ...
# all applications use these settings and services
domain: shared-domain.example.com
memory: 1G
instances: 1
services:
- clockwork-mysql
applications:
- name: springtock
  host: tock09876
  path: ./spring-music/build/libs/spring-music.war
- name: springtick
  host: tick09875
  path: ./spring-music/build/libs/spring-music.war
```

In the next section we carry this principle further by distributing content across multiple manifests.

Multiple Manifests with Inheritance

A single manifest can describe multiple applications. Another powerful technique is to create multiple manifests with inheritance. Here, manifests have parent-child relationships such that children inherit descriptions from a parent. Children can use inherited descriptions as-is, extend them, or override them.

Content in the child manifest overrides content in the parent manifest, if the two conflict.

This technique helps in these and other scenarios:

- An application has a set of different deployment modes, such as debug, local, and public. Each deployment mode is described in child manifests that extend the settings in a base parent manifest.
- An application is packaged with a basic configuration described by a parent manifest. Users can extend the basic configuration by creating child manifests that add new properties or override those in the parent manifest.

The benefits of multiple manifests with inheritance are similar to those of minimizing duplicated content within single manifests. With inheritance, though, we “promote” content by placing it in the parent manifest.

Every child manifest must contain an “inherit” line that points to the parent manifest. Place the inherit line immediately after the three dashes at the top of the child manifest. For example, every child of a parent manifest named `base-manifest.yml` begins as follows:

```
---
inherit: base-manifest.yml
...
```

You do not need to add anything to the parent manifest.

In the simple example below, a parent manifest gives each application minimal resources, while a production child manifest scales them up.

simple-base-manifest.yml

```
---
path: .
domain: shared-domain.example.com
memory: 256M
instances: 1
services:
- singular-backend

# app-specific configuration
applications:
 - name: springtock
   host: 765shower
   path: ./april/build/libs/april-weather.war
 - name: wintertick
   host: 321flurry
   path: ./december/target/december-weather.war
```

simple-prod-manifest.yml

```
---
inherit: simple-base-manifest.yml
applications:
 - name:springstorm
   memory: 512M
   instances: 1
   host: 765deluge
   path: ./april/build/libs/april-weather.war
 - name: winterblast
   memory: 1G
   instances: 2
   host: 321blizzard
   path: ./december/target/december-weather.war
```

Note: Inheritance can add an additional level of complexity to manifest creation and maintenance. Comments that precisely explain how the child manifest extends or overrides the descriptions in the parent manifest can alleviate this complexity.

Using Application Health Checks

Page last updated:

This topic describes how to configure health checks for your applications in Cloud Foundry.

Overview

An application health check is a monitoring process that continually checks the status of a running Cloud Foundry app.

Developers can configure a health check for an application using the Cloud Foundry Command Line Interface (cf CLI) or by specifying the `health-check-http-endpoint` and `health-check-type` fields in an application manifest.

To configure a health check using the cf CLI, follow the instructions in the Configure Health Checks section below. For more information about using an application manifest to configure a health check, see the health-check-http-endpoint and health-check-type sections of the *Deploying with Application Manifest* topic.

Application health checks function as part of the app lifecycle managed by Diego architecture.

Configure Health Checks

To configure a health check while creating or updating an application, use the cf push command:

```
$ cf push YOUR-APP -u HEALTH-CHECK-TYPE -t HEALTH-CHECK-TIMEOUT
```

Replace the placeholders in the example command above as follows:

- `HEALTH-CHECK-TYPE`: Valid health check types are `port`, `process`, and `http`. See the Health Check Types section below for more information.
- `HEALTH-CHECK-TIMEOUT`: The timeout is the amount of time allowed to elapse between starting up an application and the first healthy response. See the Health Check Timeouts section for more information.

> **Note**: The health check configuration you provide with `cf push` overrides any configuration in the application manifest.

To configure a health check for an existing application or to add a custom HTTP endpoint, use the cf set-health-check command:

```
$ cf set-health-check YOUR-APP HEALTH-CHECK-TYPE --endpoint CUSTOM-HTTP-ENDPOINT
```

Replace the placeholders in the example command above as follows:

- `HEALTH-CHECK-TYPE`: Valid health check types are `port`, `process`, and `http`. See the Health Check Types section below for more information.
- `CUSTOM-HTTP-ENDPOINT`: A `http` health check defaults to using `/` as its endpoint, but you can specify a custom endpoint. See the Health Check HTTP Endpoints section below for more information.

> **Note**: You can change the health check configuration of a deployed app with `cf set-health-check`, but you must restart the app for the changes to take effect.

Understand Health Checks

Health Check Lifecycle

The following table describes how application health checks work in Cloud Foundry.

Stage	Description
1	Application developer deploys an app to Cloud Foundry.

2	When deploying the app, the developer specifies a health check type for the app and, optionally, a timeout. If the developer does not specify a health check type, then the monitoring process defaults to a `port` health check.
3	Cloud Controller stages, starts, and runs the app.
4	Based on the type specified for the app, Cloud Controller configures a health check that runs periodically for each app instance.
5	When Diego starts an app instance, the application health check runs every 2 seconds until a response indicates that the app instance is healthy or until the health check timeout elapses. The 2-second health check interval is not configurable.
6	When an app instance becomes healthy, its route is advertised, if applicable. Subsequent health checks are run every 30 seconds once the app becomes healthy. The 30-second health check interval is not configurable.
7	If a previously healthy app instance fails a health check, Diego considers that particular instance to be unhealthy. As a result, Diego stops and deletes the app instance, then reschedules a new app instance. This stoppage and deletion of the app instance is reported back to the Cloud Controller as a crash event.
8	When an app instance crashes, Diego immediately attempts to restart the app instance several times. After three failed restarts, Cloud Foundry waits 30 seconds before attempting another restart. The wait time doubles each restart until the ninth restart, and remains at that duration until the 200th restart. After the 200th restart, Cloud Foundry stops trying to restart the app instance.

Health Check Types

The following table describes the types of health checks available for applications and recommended circumstances in which to use them:

Health Check Type	Recommended Use Case	Explanation
`http`	The app can provide an `HTTP 200` response.	The `http` health check performs a GET request to the configured HTTP endpoint on the app's default port. When the health check receives an `HTTP 200` response, the app is declared healthy. We recommend using the `http` health check type whenever possible. A healthy HTTP response ensures that the web app is ready to serve HTTP requests. The configured endpoint must respond within 1 second to be considered healthy.
`port`	The app can receive TCP connections (including HTTP web applications).	A health check makes a TCP connection to the port or ports configured for the app. For applications with multiple ports, a health check monitors each port. If you do not specify a health check type for your app, then the monitoring process defaults to a `port` health check. The TCP connection must be established within 1 second to be considered healthy.
`process`	The app does not support TCP connections (for example, a worker).	For a `process` health check, Diego ensures that any process declared for the app stays running. If the process exits, Diego stops and deletes the app instance.

Health Check Timeouts

The value configured for the health check timeout is the amount of time allowed to elapse between starting up an app and the first healthy response from the app. If the health check does not receive a healthy response within the configured timeout, then the app is declared unhealthy.

In Pivotal Cloud Foundry, the default timeout is 60 seconds and the maximum configurable timeout is 600 seconds.

Health Check HTTP Endpoints

Only used by `http` type, the `--endpoint` flag of the `cf set-health-check` command specifies the path portion of a URI that must be served by the app and return `HTTP 200` when the app is healthy.

Note: This command will only check the health of the default port of the app.

Note: For HTTP apps, we recommend setting the health check type to `http` instead of a simple port check.

Pivotal

Scaling an Application Using cf scale

Page last updated:

Factors such as user load, or the number and nature of tasks performed by an application, can change the disk space and memory the application uses. For many applications, increasing the available disk space or memory can improve overall performance. Similarly, running additional instances of an application can allow the application to handle increases in user load and concurrent requests. These adjustments are called **scaling** an application.

Use cf scale to scale your application up or down to meet changes in traffic or demand.

> **Note**: You can configure your app to scale automatically based on rules that you set. See the Scaling an Application Using Autoscaler topic for more information.

Scaling Horizontally

Horizontally scaling an application creates or destroys instances of your application.

Incoming requests to your application are automatically load balanced across all instances of your application, and each instance handles tasks in parallel with every other instance. Adding more instances allows your application to handle increased traffic and demand.

Use `cf scale APP -i INSTANCES` to horizontally scale your application. Cloud Foundry will increase or decrease the number of instances of your application to match `INSTANCES`.

```
$ cf scale myApp -i 5
```

Scaling Vertically

Vertically scaling an application changes the disk space limit or memory limit that Cloud Foundry applies to all instances of the application.

Use `cf scale APP -k DISK` to change the disk space limit applied to all instances of your application. `DISK` must be an integer followed by either an **M**, for megabytes, or **G**, for gigabytes.

```
$ cf scale myApp -k 512M
```

Use `cf scale APP -m MEMORY` to change the memory limit applied to all instances of your application. `MEMORY` must be an integer followed by either an **M**, for megabytes, or **G**, for gigabytes.

```
$ cf scale myApp -m 1G
```

Running Tasks

Page last updated:

This topic describes how to run tasks in Cloud Foundry. A task is an application or script whose code is included as part of a deployed application, but runs independently in its own container.

About Tasks

In contrast to a long running process (LRP), tasks run for a finite amount of time, then stop. Tasks run in their own containers and are designed to use minimal resources. After a task runs, Cloud Foundry destroys the container running the task.

As a single-use object, a task can be checked for its state and for a success or failure message.

> **Note**: Running a task consumes an application instance, and will be billed accordingly.

Use Cases for Tasks

Tasks are used to perform one-off jobs, which include the following:

- Migrating a database
- Sending an email
- Running a batch job
- Running a data processing script
- Processing images
- Optimizing a search index
- Uploading data
- Backing-up data
- Downloading content

How Tasks Are Run

Tasks are always executed asynchronously, meaning that they run independently from the parent application or other tasks that run on the same application.

The life-cycle of a task is as follows:

1. A user initiates a task in Cloud Foundry using one of the following mechanisms:
    - `cf run-task APPNAME "TASK"` command. See the Running Tasks section of this topic for more information.
    - Cloud Controller v3 API call. See the Tasks API reference page for more information.
    - Cloud Foundry Java Client. See the Cloud Foundry Java Client Library and Cloud Foundry Java Client topics for more information.

2. Cloud Foundry creates a container specifically for the task.

3. Cloud Foundry runs the task on the container using the value passed to the `cf run-task` command.

4. Cloud Foundry destroys the container.

The container also inherits environment variables, service bindings, and security groups bound to the application.

> **Note**: You cannot SSH into the container running a task.

Task Logging and Execution History

Any data or messages the task outputs to STDOUT or STDERR is available on the app's firehose logs. A syslog drain attached to the app receives the task log output.

The task execution history is retained for one month.

Manage Tasks

At the system level, a user with admin-level privileges can use the Cloud Controller v3 API to view all tasks that are running within an org or space. For more information, see the documentation for the Cloud Controller v3 API.

In addition, admins can set the default memory and disk usage quotas for tasks on a global level. Initially, tasks use the same memory and disk usage defaults as applications. However, the default memory and disk allocations for tasks can be defined separately from the default app memory and disk allocations.

The default memory and disk allocations are defined using the **Default App Memory** and **Default Disk Quota per App** fields. These fields are available in the **Application Developer Controls** configuration screen of Pivotal Application Service (PAS).

Run a Task on an Application

You can use the Cloud Foundry Command Line Interface (cf CLI) to run a task in the context of an application.

Note: To run tasks with the cf CLI, you must install cf CLI v6.23.0 or later. See the Installing the Cloud Foundry Command Line Interface topic for information about downloading, installing, and uninstalling the cf CLI.

To run a task on an application, perform the following steps:

1. Push your application:

```
$ cf push APP-NAME
```

2. Run your task on the deployed application:

```
$ cf run-task APP-NAME "TASK" --name TASK-NAME
```

The following example runs a database migration as a task on the `my-app` application:

```
$ cf run-task my-app "bin/rails db:migrate" --name my-task
Creating task for app my-app in org jdoe-org / space development as jdoe@pivotal.io...
OK
Task 1 has been submitted successfully for execution.
```

Note: To re-run a task, you must run it as a new task using the above command.

Use the `cf logs APP-NAME --recent` command to display the recent logs of the application and all its tasks.

The following example displays the logs of a successful task:

```
$ cf logs my-app --recent
2017-01-03T15:58:06.57-0800 [APP/TASK/my-task/0]OUT Creating container
2017-01-03T15:58:08.45-0800 [APP/TASK/my-task/0]OUT Successfully created container
2017-01-03T15:58:13.32-0800 [APP/TASK/my-task/0]OUT D, [2017-01-03T23:58:13.322258 #7] DEBUG -- :    (15.9ms)  CREATE TABLE "schema_migrations" ("version" character varying
2017-01-03T15:58:13.33-0800 [APP/TASK/my-task/0]OUT D, [2017-01-03T23:58:13.337723 #7] DEBUG -- :    (11.9ms)  CREATE TABLE "ar_internal_metadata" ("key" character varying P
2017-01-03T15:58:13.34-0800 [APP/TASK/my-task/0]OUT D, [2017-01-03T23:58:13.340234 #7] DEBUG -- :    (1.6ms)  SELECT pg_try_advisory_lock(3720865444824511725);
2017-01-03T15:58:13.35-0800 [APP/TASK/my-task/0]OUT D, [2017-01-03T23:58:13.351853 #7] DEBUG -- :   ActiveRecord::SchemaMigration Load (0.7ms)  SELECT "schema_migrations".
2017-01-03T15:58:13.35-0800 [APP/TASK/my-task/0]OUT I, [2017-01-03T23:58:13.357294 #7]  INFO -- : Migrating to CreateArticles (20161118225627)
2017-01-03T15:58:13.35-0800 [APP/TASK/my-task/0]OUT D, [2017-01-03T23:58:13.359565 #7] DEBUG -- :    (0.5ms)  BEGIN
2017-01-03T15:58:13.35-0800 [APP/TASK/my-task/0]OUT == 20161118225627 CreateArticles: migrating ===================================
2017-01-03T15:58:13.50-0800 [APP/TASK/my-task/0]OUT Exit status 0
2017-01-03T15:58:13.56-0800 [APP/TASK/my-task/0]OUT Destroying container
2017-01-03T15:58:15.65-0800 [APP/TASK/my-task/0]OUT Successfully destroyed container
```

The following example displays the logs of a failed task:

```
$ cf logs my-app --recent
2016-12-14T11:09:26.09-0800 [APP/TASK/my-task/0]OUT Creating container
2016-12-14T11:09:28.43-0800 [APP/TASK/my-task/0]OUT Successfully created container
2016-12-14T11:09:28.85-0800 [APP/TASK/my-task/0]ERR bash: bin/rails: command not found
2016-12-14T11:09:28.85-0800 [APP/TASK/my-task/0]OUT Exit status 127
2016-12-14T11:09:28.89-0800 [APP/TASK/my-task/0]OUT Destroying container
2016-12-14T11:09:30.50-0800 [APP/TASK/my-task/0]OUT Successfully destroyed container
```

If your task name is unique, you can `grep` the output of the `cf logs` command for the task name to view task-specific logs.

List Tasks Running on an Application

To list the tasks for a given application, run the `cf tasks APP-NAME`. For example:

```
$ cf tasks my-app
Getting tasks for app my-app in org jdoe-org / space development as jdoe@pivotal.io...
OK

id   name       state       start time                      command
2    339044ef   FAILED      Wed, 23 Nov 2016 21:52:52 UTC   echo foo; sleep 100; echo bar
1    8d0618cf   SUCCEEDED   Wed, 23 Nov 2016 21:37:28 UTC   bin/rails db:migrate
```

Each task has one of the following states:

State	Description
RUNNING	The task is currently in progress.
FAILED	The task did not complete. This state occurs when a task does not work correctly or a user cancels the task.
SUCCEEDED	The task completed successfully.

Cancel a Task

After running a task, you may be able to cancel it before it finishes. To cancel a running task, use the `cf terminate-task APP-NAME TASK-ID` command. For example:

```
$ cf terminate-task my-app 2
Terminating task 2 of app my-app in org jdoe-org / space development as jdoe@pivotal.io...
OK
```

Pivotal

Cloud Foundry Environment Variables

Page last updated:

Environment variables are the means by which the Cloud Foundry (CF) runtime communicates with a deployed application about its environment. This page describes the environment variables that the runtime and buildpacks set for applications.

For information about setting your own application-specific environment variables, see the Environment Variable section of the *Deploying with Application Manifests* topic.

View Environment Variables

Install the Cloud Foundry Command Line Interface (cf CLI), and use the cf env command to view the Cloud Foundry environment variables for your application. The `cf env` command displays the following environment variables:

- The `VCAP_APPLICATION` and `VCAP_SERVICES` variables provided in the container environment
- The user-provided variables set using the cf set-env command

```
$ cf env my-app
Getting env variables for app my-app in org my-org / space my-space as
admin...
OK
System-Provided:

{
 "VCAP_APPLICATION": {
  "application_id": "fa05c1a9-0fc1-4fbd-bae1-139850dec7a3",
  "application_name": "my-app",
  "application_uris": [
   "my-app.192.0.2.34.xip.io"
  ],
  "application_version": "fb8fbcc6-8d58-479e-bcc7-3b4ce5a7f0ca",
  "cf_api": "https://api.example.com",
  "limits": {
   "disk": 1024,
   "fds": 16384,
   "mem": 256
  },
  "name": "my-app",
  "space_id": "06450c72-4669-4dc6-8096-45f9777db68a",
  "space_name": "my-space",
  "uris": [
   "my-app.192.0.2.34.xip.io"
  ],
  "users": null,
  "version": "fb8fbcc6-8d58-479e-bcc7-3b4ce5a7f0ca"
 }

User-Provided:
MY_DRAIN: http://drain.example.com
MY_ENV_VARIABLE: 100
```

Application-Specific System Variables

The subsections that follow describe the environment variables that Cloud Foundry makes available to your application container. Some of these variables are the same across instances of a single application, and some vary from instance to instance.

You can access environment variables programmatically, including variables defined by the buildpack. For more information, refer to the buildpack documentation for Java, Node.js, and Ruby.

Env Var	Running	Staging	Task
`CF_INSTANCE_ADDR`	x	x	x
`CF_INSTANCE_GUID`	x		x
`CF_INSTANCE_INDEX`	x		
`CF_INSTANCE_INTERNAL_IP`	x	x	x

`CF_INSTANCE_IP`	x	x	x
`CF_INSTANCE_PORT`	x	x	x
`CF_INSTANCE_PORTS`	x	x	x
`CF_STACK`		x	
`DATABASE_URL`	x		x
`HOME`	x	x	x
`INSTANCE_GUID`	x		
`INSTANCE_INDEX`	x		
`LANG`	x	x	x
`MEMORY_LIMIT`	x	x	x
`PATH`	x	x	x
`PORT`	x		
`PWD`	x	x	x
`TMPDIR`	x		x
`USER`	x	x	x
`VCAP_APP_HOST`	x		
`VCAP_APP_PORT`	x		
`VCAP_APPLICATION`	x	x	x
`VCAP_SERVICES`	x	x	x

CF_INSTANCE_ADDR

The CF_INSTANCE_IP and CF_INSTANCE_PORT of the app instance in the format `IP:PORT`.

Example: `CF_INSTANCE_ADDR=1.2.3.4:5678`

CF_INSTANCE_GUID

The UUID of the particular instance of the app.

Example: `CF_INSTANCE_GUID=41653aa4-3a3a-486a-4431-ef258b39f042`

CF_INSTANCE_INDEX

The index number of the app instance.

Example: `CF_INSTANCE_INDEX=0`

CF_INSTANCE_IP

The external IP address of the host running the app instance.

Example: `CF_INSTANCE_IP=1.2.3.4`

CF_INSTANCE_INTERNAL_IP

The internal IP address of the container running the app instance.

Example: `CF_INSTANCE_INTERNAL_IP=5.6.7.8`

CF_INSTANCE_PORT

The external, or *host-side*, port corresponding to the internal, or *container-side*, port with value PORT. This value is generally different from the PORT of the app instance.

Example: `CF_INSTANCE_PORT=61045`

CF_INSTANCE_PORTS

The list of mappings between internal, or *container-side*, and external, or *host-side*, ports allocated to the instance's container. Not all of the internal ports are necessarily available for the application to bind to, as some of them may be used by system-provided services that also run inside the container. These internal and external values may differ.

Example: `CF_INSTANCE_PORTS=[{external:61045,internal:8080},{external:61046,internal:2222}]`

DATABASE_URL

At runtime, CF creates a `DATABASE_URL` environment variable for every app based on the VCAP_SERVICES environment variable.

CF uses the structure of the `VCAP_SERVICES` environment variable to populate `DATABASE_URL`. CF recognizes any service containing a JSON object with the following form as a candidate for `DATABASE_URL` and uses the first candidate it finds.

```
{
 "some-service": [
   {
     "credentials": {
       "uri": "SOME-DATABASE-URL"
     }
   }
 ]
}
```

For example, consider the following `VCAP_SERVICES`:

```
VCAP_SERVICES =
{
 "elephantsql": [
   {
     "name": "elephantsql-c6c60",
     "label": "elephantsql",
     "credentials": {
       "uri": "postgres://exampleuser:examplepass@babar.elephantsql.com:5432/exampledb"
     }
   }
 ]
}
```

Based on this `VCAP_SERVICES`, CF creates the following `DATABASE_URL` environment variable:

```
DATABASE_URL = postgres://exampleuser:examplepass@babar.elephantsql.com:5432/exampledb
```

HOME

Root folder for the deployed application.

Example: `HOME=/home/vcap/app`

LANG

LANG is required by buildpacks to ensure consistent script load order.

Example: `LANG=en_US.UTF-8`

MEMORY_LIMIT

The maximum amount of memory that each instance of the application can consume. You specify this value in an application manifest or with the cf CLI when pushing an application. The value is limited by space and org quotas.

If an instance exceeds the maximum limit, it will be restarted. If Cloud Foundry is asked to restart an instance too frequently, the instance will instead be terminated.

Example: `MEMORY_LIMIT=512M`

PORT

The port on which the app should listen for requests. The Cloud Foundry runtime allocates a port dynamically for each instance of the application, so code that obtains or uses the app port should refer to it using the `PORT` environment variable.

Example: `PORT=8080`

PWD

Identifies the present working directory, where the buildpack that processed the application ran.

Example: `PWD=/home/vcap/app`

TMPDIR

Directory location where temporary and staging files are stored.

Example: `TMPDIR=/home/vcap/tmp`

USER

The user account under which the application runs.

Example: `USER=vcap`

VCAP_APP_PORT

Deprecated name for the PORT variable defined above.

VCAP_APPLICATION

This variable contains the associated attributes for a deployed application. Results are returned in JSON format. The table below lists the attributes that are returned.

Attribute	Description
`application_id`	GUID identifying the application.
`application_name`	The name assigned to the application when it was pushed.
`application_uris`	The URIs assigned to the application.
`application_version`	GUID identifying a version of the application. Each time an application is pushed or restarted, this value is updated.
`cf_api`	Location of the Cloud Controller API for the CF Deployment where the app runs.
`host`	Deprecated. IP address of the application instance.
`limits`	The limits to disk space, number of files, and memory permitted to the app. Memory and disk space limits are supplied when the application is deployed, either on the command line or in the application manifest. The number of files allowed is operator-defined.

`name`	Identical to `application_name`.
`space_id`	GUID identifying the application's space.
`space_name`	Human-readable name of the space where the app is deployed.
`start`	Human-readable timestamp for the time the instance was started. Not provided on Diego Cells.
`started_at`	Identical to `start`. Not provided on Diego Cells.
`started_at_timestamp`	Unix epoch timestamp for the time the instance was started. Not provided on Diego Cells.
`state_timestamp`	Identical to `started_at_timestamp`. Not provided on Diego Cells.
`uris`	Identical to `application_uris`. You must ensure that both `application_uris` and `uris` are set to the same value.
`users`	Deprecated. Not provided on Diego Cells.
`version`	Identical to `application_version`.

The following example shows how to set the `VCAP_APPLICATION` environment variable:

```
VCAP_APPLICATION={"instance_id":"fe98dc76ba549876543210abcd1234",
"instance_index":0,"host":"0.0.0.0","port":61857,"started_at":"2013-08-12
00:05:29 +0000","started_at_timestamp":1376265929,"start":"2013-08-12 00:05:29
+0000","state_timestamp":1376265929,"limits":{"mem":512,"disk":1024,"fds":16384}
,"application_version":"ab12cd34-5678-abcd-0123-abcdef987654","application_name"
:"styx-james","application_uris":["my-app.example.com"],"version":"ab1
2cd34-5678-abcd-0123-abcdef987654","name":"my-app","uris":["my-app.example.com"]
,"users":null}
```

VCAP_SERVICES

For bindable services, Cloud Foundry adds connection details to the `VCAP_SERVICES` environment variable when you restart your application, after binding a service instance to your application.

The results are returned as a JSON document that contains an object for each service for which one or more instances are bound to the application. The service object contains a child object for each service instance of that service that is bound to the application. The attributes that describe a bound service are defined in the table below.

The key for each service in the JSON document is the same as the value of the "label" attribute.

Attribute	Description
`name`	The name assigned to the service instance by the user.
`label`	The name of the service offering.
`tags`	An array of strings an app can use to identify a service instance.
`plan`	The service plan selected when the service instance was created.
`credentials`	A JSON object containing the service-specific credentials needed to access the service instance.

To see the value of VCAP_SERVICES for an application pushed to Cloud Foundry, see View Environment Variable Values.

The example below shows the value of VCAP_SERVICES for bound instances of several services available in the Pivotal Web Services Marketplace.

```
VCAP_SERVICES=
{
 "elephantsql": [
  {
   "name": "elephantsql-c6c60",
   "label": "elephantsql",
   "tags": [
    "postgres",
    "postgresql",
    "relational"
   ],
   "plan": "turtle",
   "credentials": {
    "uri": "postgres://exampleuser:examplepass@babar.elephantsql.com:5432/exampleuser"
   }
  }
 ],
 "sendgrid": [
  {
   "name": "mysendgrid",
   "label": "sendgrid",
   "tags": [
    "smtp"
   ],
   "plan": "free",
   "credentials": {
    "hostname": "smtp.sendgrid.net",
    "username": "QvsXMbJ3rK",
    "password": "HCHMOYluTv"
   }
  }
 ]
}
```

Environment Variable Groups

Environment variable groups are system-wide variables that enable operators to apply a group of environment variables to all running applications and all staging applications separately.

An environment variable group consists of a single hash of name-value pairs that are later inserted into an application container at runtime or at staging. These values can contain information such as HTTP proxy information. The values for variables set in an environment variable group are case-sensitive.

When creating environment variable groups, consider the following:

- Only the Cloud Foundry operator can set the hash value for each group.
- All authenticated users can get the environment variables assigned to their application.
- All variable changes take effect after the operator restarts or restages the applications.
- Any user-defined variable takes precedence over environment variables provided by these groups.

The table below lists the commands for environment variable groups.

CLI Command	Description
`running-environment-variable-group` or `revg`	Retrieves the contents of the running environment variable group.
`staging-environment-variable-group` or `sevg`	Retrieves the contents of the staging environment variable group.
`set-staging-environment-variable-group` or `ssevg`	Passes parameters as JSON to create a staging environment variable group.
`set-running-environment-variable-group` or `srevg`	Passes parameters as JSON to create a running environment variable group.

The following examples demonstrate how to retrieve the environment variables:

```
$ cf revg
Retrieving the contents of the running environment variable group as
sampledeveloper@example.com...
OK
Variable Name   Assigned Value
HTTP Proxy      198.51.100.130

$ cf sevg
Retrieving the contents of the staging environment variable group as
sampledeveloper@example.com...
OK
Variable Name   Assigned Value
HTTP Proxy      203.0.113.105
EXAMPLE-GROUP   2001

$ cf apps
Getting apps in org SAMPLE-ORG-NAME / space dev as
sampledeveloper@example.com...
OK

name    requested state   instances   memory   disk   urls
my-app  started           1/1         256M     1G     my-app.com

$ cf env APP-NAME
Getting env variables for app APP-NAME in org SAMPLE-ORG-NAME / space dev as
sampledeveloper@example.com...
OK

System-Provided:

{
 "VCAP_APPLICATION": {
 "application_name": "APP-NAME",
 "application_uris": [
  "my-app.example.com"
 ],
 "application_version": "7d0d64be-7f6f-406a-9d21-504643147d63",
 "limits": {
 "disk": 1024,
 "fds": 16384,
 "mem": 256
 },
 "name": "APP-NAME",
 "space_id": "37189599-2407-9946-865e-8ebd0e2df89a",
 "space_name": "dev",
 "uris": [
  "my-app.example.com"
 ],
 "users": null,
 "version": "7d0d64be-7f6f-406a-9d21-504643147d63"
 }
}

Running Environment Variable Groups:
HTTP Proxy: 198.51.100.130

Staging Environment Variable Groups:
EXAMPLE-GROUP: 2001
HTTP Proxy: 203.0.113.105
```

The following examples demonstrate how to set environment variables:

```
$ cf ssevg '{"test":"198.51.100.130","test2":"203.0.113.105"}'
Setting the contents of the staging environment variable group as admin...
OK
$ cf sevg
Retrieving the contents of the staging environment variable group as admin...
OK
Variable Name   Assigned Value
test            198.51.100.130
test2           203.0.113.105

$ cf srevg '{"test3":"2001","test4":"2010"}'
Setting the contents of the running environment variable group as admin...
OK
$ cf revg
Retrieving the contents of the running environment variable group as admin...
OK
Variable Name   Assigned Value
test3           2001
test4           2010
```

Using Blue-Green Deployment to Reduce Downtime and Risk

Page last updated:

Blue-green deployment is a technique that reduces downtime and risk by running two identical production environments called Blue and Green.

At any time, only one of the environments is live, with the live environment serving all production traffic. For this example, Blue is currently live and Green is idle.

As you prepare a new version of your software, deployment and the final stage of testing takes place in the environment that is *not* live: in this example, Green. Once you have deployed and fully tested the software in Green, you switch the router so all incoming requests now go to Green instead of Blue. Green is now live, and Blue is idle.

This technique can eliminate downtime due to application deployment. In addition, blue-green deployment reduces risk: if something unexpected happens with your new version on Green, you can immediately roll back to the last version by switching back to Blue.

Note: If your app uses a relational database, blue-green deployment can lead to discrepancies between your Green and Blue databases during an update. To maximize data integrity, configure a single database for backward and forward compatibility.

Note: You can adjust the route mapping pattern to display a static maintenance page during a maintenance window for time-consuming tasks, such as migrating a database. In this scenario, the router switches all incoming requests from Blue to Maintenance to Green.

Blue-Green Deployment with Cloud Foundry Example

For this example, we'll start with a simple application: "demo-time." This app is a web page that displays the words "Blue time" and the date/time on the server.

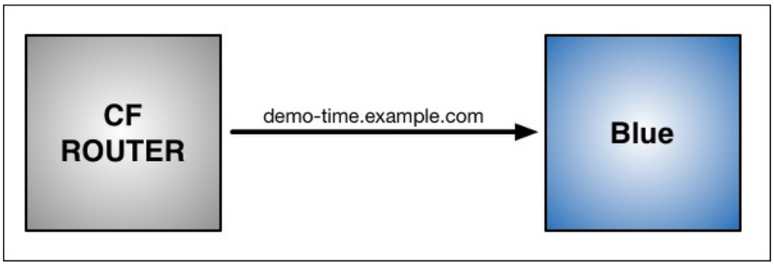
Step 1: Push an App

Use the Cloud Foundry Command Line Interface (cf CLI) to push the application. Name the application "Blue" with the subdomain "demo-time."

```
$ cf push Blue -n demo-time
```

As shown in the graphic below:

- Blue is now running on Cloud Foundry.
- The CF Router sends all traffic for `demo-time.example.com` traffic to Blue.



Step 2: Update App and Push

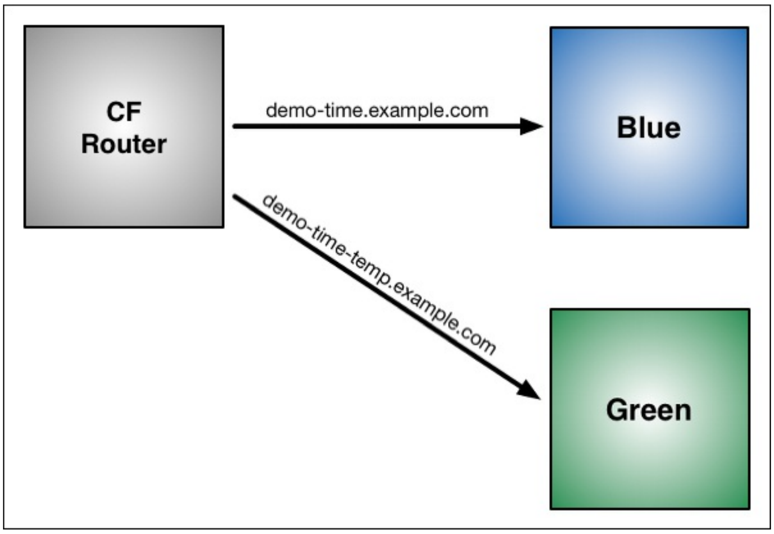
Now make a change to the application. First, replace the word "Blue" on the web page with "Green," then rebuild the source file for the application. Run `cf push` again, but use the name "Green" for the application and provide a different subdomain to create a temporary route:

```
$ cf push Green -n demo-time-temp
```

After this push:

- Two instances of our application are now running on Cloud Foundry: the original Blue and the updated Green.

- The CF Router continues sending all traffic for `demo-time.example.com` to Blue. The router now also sends any traffic for `demo-time-temp.example.com` to Green.



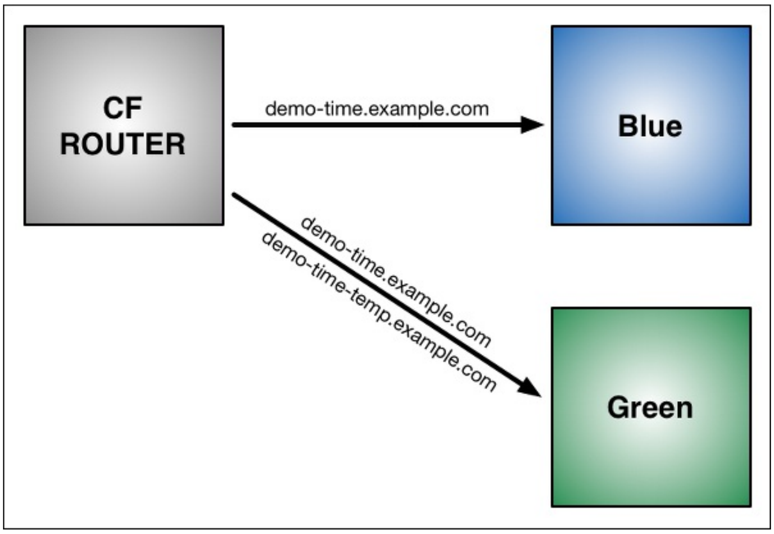
Step 3: Map Original Route to Green

Now that both apps are up and running, switch the router so all incoming requests go to the Green app *and* the Blue app. Do this by mapping the original URL route (`demo-time.example.com`) to the Green application using the cf map-route command.

```
$ cf map-route Green example.com -n demo-time
Binding demo-time.example.com to Green... OK
```

After the `cf map-route` command :

- The CF Router continues sending traffic for `demo-time-temp.example.com` to Green.
- Within a few seconds, the CF Router begins load balancing traffic for `demo-time.example.com` between Blue and Green.



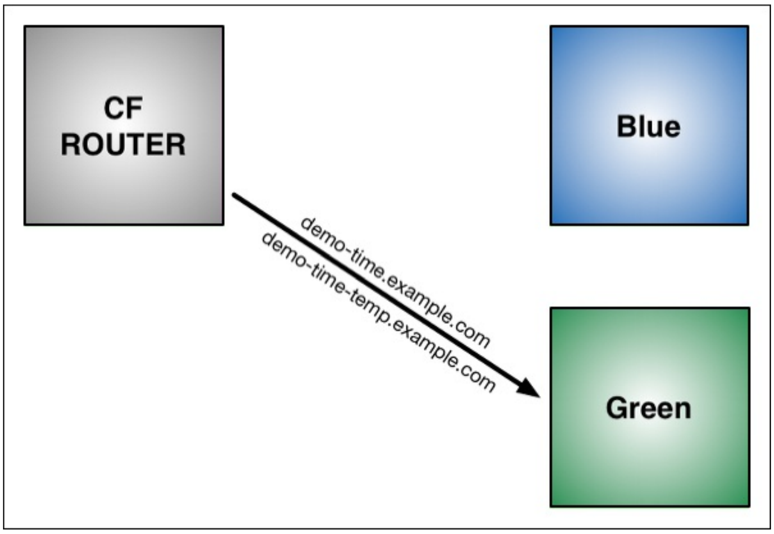
Step 4: Unmap Route to Blue

Once you verify Green is running as expected, stop routing requests to Blue using the cf unmap-route command:

```
$ cf unmap-route Blue example.com -n demo-time
Unbinding demo-time.example.com from blue... OK
```

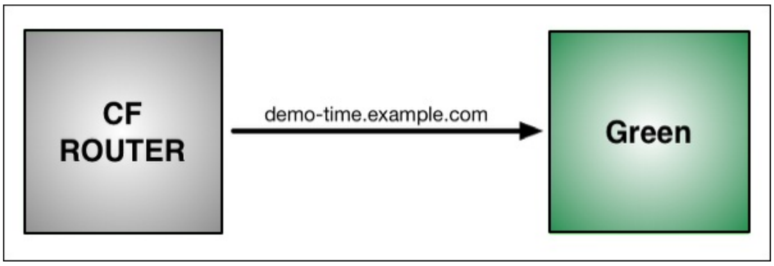
After `cf unmap-route` command:

- The CF Router stops sending traffic to Blue. Now all traffic for `demo-time.example.com` is sent to Green:



Step 5: Remove Temporary Route to Green

You can now use `cf unmap-route` to remove the route `demo-time-temp.example.com` from Green. The route can be deleted using `cf delete-route` or reserved for later use. You can also decommission Blue, or keep it in case you need to roll back your changes.



Implementations

Cloud Foundry community members have written plugins to automate the blue-green deployment process. These include:

- Autopilot: Autopilot is a Cloud Foundry Go plugin that provides a subcommand, `zero-downtime-push`, for hands-off, zero-downtime application deploys.
- BlueGreenDeploy: cf-blue-green-deploy is a plugin, written in Go, for the Cloud Foundry Command Line Interface (cf CLI) that automates a few steps involved in zero-downtime deploys.

Troubleshooting Application Deployment and Health

Page last updated:

Refer to this topic for help diagnosing and resolving common issues when you deploy and run applications on Cloud Foundry.

Common Issues

The following sections describe common issues you might encounter when attempting to deploy and run your application, and possible resolutions.

cf push Times Out

If your deployment times out during the upload or staging phase, you may receive one of the following error messages:

- `504 Gateway Timeout`
- `Error uploading application`
- `Timed out waiting for async job JOB-NAME to finish`

If this happens, do the following:

- **Check your network speed.** Depending on the size of your application, your `cf push` could be timing out because the upload is taking too long. We recommended an Internet connection speed of at least 768 KB/s (6 Mb/s) for uploads.
- **Make sure you are pushing only needed files.** By default, `cf push` will push all the contents of the current working directory. Make sure you are pushing only the directory for your application. If your application is too large, or if it has many small files, Cloud Foundry may time out during the upload. To reduce the number of files you are pushing, ensure that you push only the directory for your application, and remove unneeded files or use the `.cfignore` file to specify excluded files.
- **Set the CF_STAGING_TIMEOUT and CF_STARTUP_TIMEOUT environment variables.** By default your app has 15 minutes to stage and 5 minutes to start. You can increase these times by setting `CF_STAGING_TIMEOUT` and `CF_STARTUP_TIMEOUT`. Type `cf push -h` at the command line for more information.
- **If your app contains a large number of files, try pushing the app repeatedly.** Each push uploads a few more files. Eventually, all files have uploaded and the push succeeds. This is less likely to work if your app has many small files.

App Too Large

If your application is too large, you may receive one of the following error messages on `cf push`:

- `413 Request Entity Too Large`
- `You have exceeded your organization's memory limit`

If this happens, do the following:

- **Make sure your org has enough memory for all instances of your app.** You will not be able to use more memory than is allocated for your organization. To view the memory quota for your org, use `cf org ORG_NAME`.
  Your total memory usage is the sum of the memory used by all applications in all spaces within the org. Each application's memory usage is the memory allocated to it multiplied by the number of instances. To view the memory usage of all the apps in a space, use `cf apps`.
- **Make sure your application is less than 1 GB.** By default, Cloud Foundry deploys all the contents of the current working directory. To reduce the number of files you are pushing, ensure that you push only the directory for your application, and remove unneeded files or use the `.cfignore` file to specify excluded files. The following limits apply:
  - The app files to push cannot exceed 1 GB.
  - The droplet that results from compiling those files cannot exceed 1.5 GB. Droplets are typically a third larger than the pushed files.
  - The combined size of the app files, compiled droplet, and buildpack cache cannot total more than 4 GB of space during staging.

Unable to Detect a Supported Application Type

If Cloud Foundry cannot identify an appropriate buildpack for your app, you will see an error message that states `Unable to detect a supported application type`

.

You can view what buildpacks are available with the `cf buildpacks` command.

If you see a buildpack that you believe should support your app, refer to the buildpack documentation for details about how that buildpack detects applications it supports.

If you do not see a buildpack for your app, you may still be able to push your application with a custom buildpack using `cf push -b` with a path to your buildpack.

App Deploy Fails

Even when the deploy fails, the app might exist on PAS. Run `cf apps` to review the apps in the currently targeted org and space. You might be able to correct the issue using the CLI or Apps Manager, or you might have to delete the app and redeploy.

Common reasons deploying an app fails include the following:

- You did not successfully create and bind a needed service instance to the app, such as a PostgreSQL or MongoDB service instance. Refer to Step 3: Create and Bind a Service Instance for a RoR Application.
- You did not successfully create a unique URL for the app. Refer to the troubleshooting tip App Requires Unique URL.

App Requires Unique URL

PAS requires that each app that you deploy has a unique URL. Otherwise, the new app URL collides with an existing app URL and PAS cannot successfully deploy the app. You can resolve this issue by running `cf push` with either of the following flags to create a unique URL:

- `-n` to assign a different HOST name for the app.
- `--random-route` to create a URL that includes the app name and random words. Using this option might create a long URL, depending on the number of words that the app name includes.

App Fails to Start

After `cf push` stages the app and uploads the droplet, the app may fail to start, commonly with a pattern of starting and crashing similar to the following example:

```
-----> Uploading droplet (23M)
...
0 of 1 instances running, 1 starting
0 of 1 instances running, 1 down
...
0 of 1 instances running, 1 failing
FAILED
Start unsuccessful
```

If this happens, try the following:

- **Find the reason app is failing and modify your code.** Run `cf events APP-NAME` and `cf logs APP-NAME --recent` and look for messages similar to this:

  ```
  2014-04-29T17:52:34.00-0700   app.crash        index: 0, reason: CRASHED,
  exit_description: app instance exited, exit_status: 1
  ```

  These messages may identify a memory or port issue. If they do, take that as a starting point when you re-examine and fix your application code.
- **Make sure your application code uses the `PORT` environment variable.** Your application may be failing because it is listening on the wrong port. Instead of hard coding the port on which your application listens, use the `PORT` environment variable.
  For example, this Ruby snippet assigns the port value to the `listen_here` variable: `listen_here = ENV['PORT']`
  For more examples specific to your application framework, see the appropriate buildpacks documentation for your app's language.
- **Make sure your app adheres to the principles of the Twelve-Factor App and Prepare to Deploy an Application.** These texts explain how to prevent situations where your app builds locally but fails to build in the cloud.
- **Verify the timeout configuration of your app.** Application health checks use a timeout setting when an app starts up for the first time. See Using

Application Health Checks. If an application fails to start up due to health check timeout, you might see messages in the logs similar to the following:

```
2017-01-30T14:07:20.39-0800 [CELL/0]     OUT Creating container
2017-01-30T14:07:20.65-0800 [CELL/0]     OUT Successfully created container
2017-01-30T14:07:22.30-0800 [CELL/0]     OUT Starting health monitoring of container
2017-01-30T14:08:23.52-0800 [CELL/0]     ERR Timed out after 1m0s:
health check never passed.
2017-01-30T14:08:23.57-0800 [CELL/0]     OUT Destroying container
2017-01-30T14:08:23.59-0800 [API/0]      OUT Process has crashed with type: "web"
2017-01-30T14:08:23.59-0800 [CELL/0]     OUT Creating container
2017-01-30T14:08:23.60-0800 [API/0]      OUT App instance exited with guid 91086440-bac0-44f0-808f-a034a1ec2ed0
payload: {"instance"=>"", "index"=>0, "reason"=>"CRASHED",
"exit_description"=>"2 error(s) occurred:\n\n* 1 error(s)
occurred:\n\n* Exited with status 6\n* 2 error(s)
occurred:\n\n* cancelled\n* cancelled", "crash_count"=>1,
"crash_timestamp"=>1485814103565763172,
"version"=>"3e6e4232-7e19-4168-9583-1176833d2c71"}
2017-01-30T14:08:23.83-0800 [CELL/0]     OUT Successfully destroyed container
2017-01-30T14:08:23.84-0800 [CELL/0]     OUT Successfully created container
2017-01-30T14:08:25.41-0800 [CELL/0]     OUT Starting health monitoring of container
```

App consumes too much memory, then crashes

An app that `cf push` has uploaded and started can crash later if it uses too much memory.

Make sure your app is not consuming more memory than it should. When you ran `cf push` and `cf scale`, that configured a limit on the amount of memory your app should use. Check your app's actual memory usage. If it exceeds the limit, modify the app to use less memory.

Routing Conflict

PAS allows multiple apps, or versions of the same app, to be mapped to the same route. This feature enables Blue-Green deployment. For more information see Using Blue-Green Deployment to Reduce Downtime and Risk.

Routing multiple apps to the same route may cause undesirable behavior in some situations by routing incoming requests randomly to one of the apps on the shared route.

If you suspect a routing conflict, run `cf routes` to check the routes in your installation.

If two apps share a route outside of a Blue-Green deploy strategy, choose one app to re-assign to a different route and follow the procedure below:

1. Run `cf unmap-route YOUR-APP-NAME OLD-ROUTE` to remove the existing route from that app.
2. Run `cf map-route YOUR-APP-NAME NEW-ROUTE` to map the app to a new, unique route.

Gather Diagnostic Information

Use the techniques in this section to gather diagnostic information and troubleshoot app deployment issues.

Examine Environment Variables

`cf push` deploys your application to a container on the server. The environment variables in the container govern your application.

You can set environment variables in a manifest created before you deploy. See Deploying with Application Manifests.

You can also set an environment variable with a `cf set-env` command followed by a `cf push` command. You must run `cf push` for the variable to take effect in the container environment.

Use the `cf env` command to view the environment variables that you have set using the `cf set-env` command and the variables in the container environment:

```
$ cf env my-app
 Getting env variables for app my-app in org My-Org / space development as admin...
 OK

 System-Provided:
 {
  "VCAP_SERVICES": {
   "p-mysql-n/a": [
    {
     "credentials": {
      "uri":"postgres://lrra:e6B-X@p-mysqlprovider.example.com:5432/lraa
     },
     "label": "p-mysql-n/a",
     "name": "p-mysql",
     "syslog_drain_url": "",
     "tags": ["postgres","postgresql","relational"]
    }
   ]
  }
 }

 User-Provided:
 my-env-var: 100
 my-drain: http://drain.example.com
```

View Logs

To view app logs streamed in real-time, use the `cf logs APP-NAME` command.

To aggregate your app logs to view log history, bind your app to a syslog drain service. For more information, see Streaming Application Logs to Log Management Services.

> **Note**: The Diego architecture does not support the `cf files` command.

Trace Cloud Controller REST API Calls

If a command fails or produces unexpected results, re-run it with HTTP tracing enabled to view requests and responses between the Cloud Foundry Command Line Interface (cf CLI) and the Cloud Controller REST API.

For example:

- Re-run `cf push` with `-v`:
  `cf push APP-NAME -v`
- Re-run `cf push` while appending API request diagnostics to a log file:
  `CF_TRACE=PATH-TO-TRACE.LOG cf push APP-NAME`

These examples enable HTTP tracing for a single command only. To enable it for an entire shell session, set the variable first:

`export CF_TRACE=true`

`export CF_TRACE=PATH-TO-TRACE.LOG`

> **Note:** `CF_TRACE` is a local environment variable that modifies the behavior of the cf CLI. Do not confuse `CF_TRACE` with the variables in the container environment where your apps run.

Analyze Zipkin Trace IDs

When the Zipkin feature is enabled in Cloud Foundry, the Gorouter adds or forwards Zipkin trace IDs and span IDs to HTTP headers. For more information about what the Gorouter provides in the HTTP header, see the HTTP Headers section of the *HTTP Routing* topic.

After adding Zipkin HTTP headers to app logs, developers can use `cf logs myapp` to correlate the trace and span ids logged by the Gorouter with the trace ids logged by their app. To correlate trace ids for a request through multiple apps, each app must forward appropriate values for the headers with requests to other applications.

Use Troubleshooting Commands

You can investigate app deployment and health using the cf CLI.

Some cf CLI commands may return connection credentials. Remove credentials and other sensitive information from command output before you post the output a public forum.

- `cf apps` : Returns a list of the applications deployed to the current space with deployment options, including the name, current state, number of instances, memory and disk allocations, and URLs of each application.
- `cf app APP-NAME` : Returns the health and status of each instance of a specific application in the current space, including instance ID number, current state, how long it has been running, and how much CPU, memory, and disk it is using.

> **Note**: CPU values returned by `cf app` show the total usage of each app instance on all CPU cores on a host VM, where each core contributes 100%. For example, the CPU of a single-threaded app instance on a Diego cell with one core cannot exceed 100%, and four instances sharing the cell cannot exceed an average CPU of 25%. A multi-threaded app instance running alone on a cell with eight cores can draw up to 800% CPU.

- `cf env APP-NAME` : Returns environment variables set using `cf set-env` and variables existing in the container environment.
- `cf events APP-NAME` : Returns information about application crashes, including error codes. Shows that an app instance exited. For more detail, look in the application logs. See https://github.com/cloudfoundry/errors for a list of Cloud Foundry errors.
- `cf logs APP-NAME --recent` : Dumps recent logs. See Viewing Logs in the Command Line Interface.
- `cf logs APP-NAME` : Returns a real-time stream of the application STDOUT and STDERR. Use **Ctrl-C** (^C) to exit the real-time stream.
- `cf files APP-NAME` : Lists the files in an application directory. Given a path to a file, outputs the contents of that file. Given a path to a subdirectory, lists the files within. Use this to explore individual logs.

> **Note**: Your application should direct its logs to STDOUT and STDERR. The `cf logs` command also returns messages from any log4j facility that you configure to send logs to STDOUT.

Access Apps with SSH

If you need to troubleshoot an instance of an app, you can gain SSH access to the app with the SSH proxy and daemon. See the Application SSH Overview topic for more information.

Send Requests to App Instance

To obtain debug data without SSH, you can make HTTP requests to a specific instance of an app by using the `X-CF-APP-INSTANCE` HTTP header. See the App Instance Routing section of the *HTTP Routing* topic for more information.

Application SSH Overview

Page last updated:

This topic introduces SSH configuration for applications in your Pivotal Application Service deployment.

If you need to troubleshoot an instance of an app, you can gain SSH access to the app using the SSH proxy and daemon.

For example, one of your app instances may be unresponsive, or the log output from the app may be inconsistent or incomplete. You can SSH into the individual VM that runs the problem instance to troubleshoot.

SSH Access Control Hierarchy

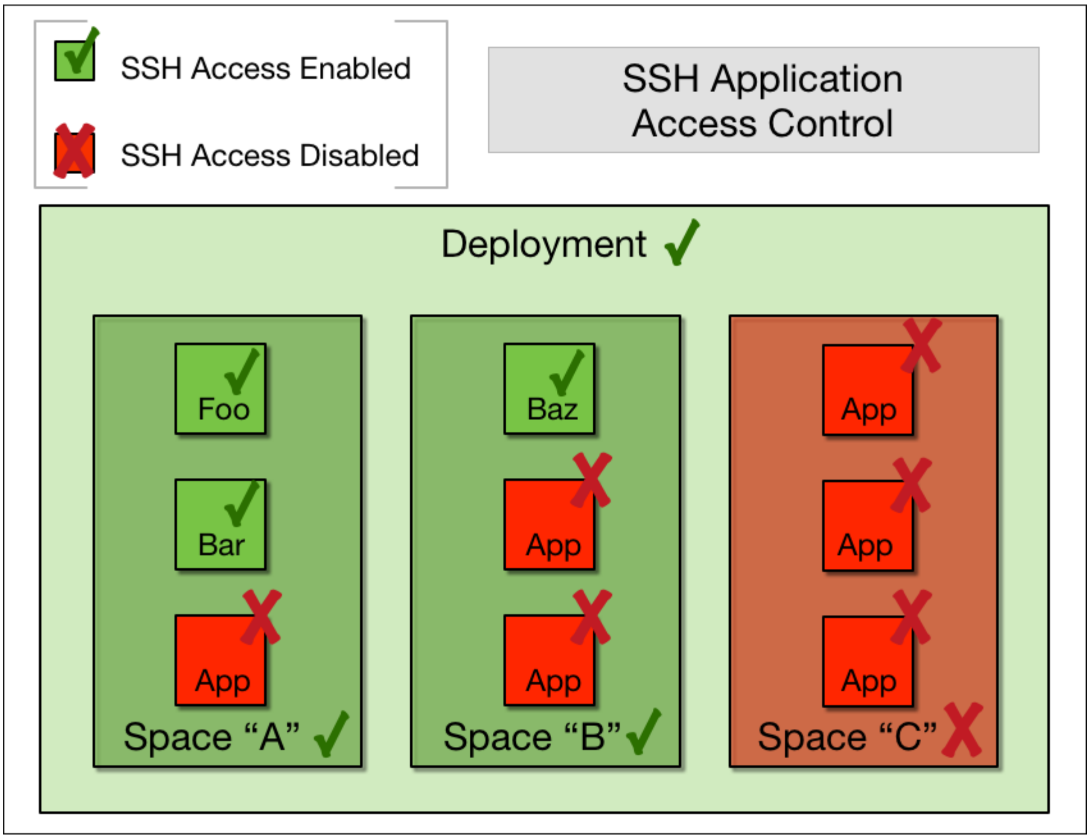
Operators, space managers, and space developers can configure SSH access for PAS, spaces, and apps as described in this table:

User Role	Scope of SSH Permissions Control	How They Define SSH Permissions
Operator	Entire deployment	Configure the deployment to allow or prohibit SSH access (one-time). For more information, see Configuring SSH Access for PCF.
Space Manager	Space	cf CLI allow-space-ssh and disallow-space-ssh commands
Space Developer	Application	cf CLI enable-ssh and disable-ssh commands

An application is SSH-accessible only if operators, space managers, and space developers all grant SSH access at their respective levels. For example, the image below shows a deployment where:

- An operator allowed SSH access at the deployment level.
- A space manager allowed SSH access for applications running in spaces “A” and “B” but not “C.”
- A space developer enabled SSH access for applications that include “Foo,” “Bar,” and “Baz.”

As a result, apps “Foo,” “Bar,” and “Baz” accept SSH requests.



SSH Access for Apps and Spaces

Space managers and space developers can configure SSH access from the command line. The Cloud Foundry Command Line Interface (cf CLI) also includes commands to return the value of the SSH access setting. See the Accessing Apps with Diego SSH topic to use and configure SSH at both the application level and the space level.

Configuring SSH Access for Pivotal Application Service

Pivotal Cloud Foundry deployments control SSH access to apps at the PAS level. Additionally, Cloud Foundry supports load balancing of SSH sessions with your load balancer. The Configuring SSH Access topic describes how to set SSH access for your deployment.

Understanding SSH Access

The SSH system components include the SSH proxy and daemon, and the system also supports authentication, and load balancing of incoming SSH traffic. The Understanding SSH topic provides a conceptual overview.

Accessing Apps with SSH

Page last updated:

This page assumes you are using cf CLI v6.13.0 or later.

The Cloud Foundry Command Line Interface (cf CLI) lets you securely log into remote host virtual machines (VMs) running Pivotal Application Service application instances. This topic describes the commands that enable SSH access to applications, and enable, disable, and check permissions for such access.

The cf CLI looks up the `app_ssh_oauth_client` identifier in the Cloud Controller `/v2/info` endpoint, and uses this identifier to query the UAA server for an SSH authorization code. On the target VM side, the SSH proxy contacts the Cloud Controller through the `app_ssh_endpoint` listed in `/v2/info` to confirm permission for SSH access.

Application SSH Commands

cf CLI Command	Purpose

`cf enable-ssh`
`cf disable-ssh`
`cf allow-space-ssh`
`cf disallow-space-ssh`
	Enable and Disable SSH Access

`cf ssh-enabled` `cf space-ssh-allowed`	Check SSH Access Permissions
`cf ssh`	Securely log into an application container
`cf ssh-code`	Enable secure log in to an application container using non-CF SSH tools like `ssh`, `scp`, and `sftp`

Enabling and Disabling SSH Access

A cloud operator can deploy Pivotal Application Service to either allow or prohibit Application SSH across the entire deployment. For more information, see Configuring SSH Access for PCF.

Within a deployment that permits SSH access to applications, Space Developers can enable or disable SSH access to individual applications, and Space Managers can enable or disable SSH access to all apps running within a space.

You must restart your application after enabling SSH access.

Configuring SSH Access at the Application Level

cf enable-ssh enables SSH access to all instances of an app:

```
$ cf enable-ssh MY-AWESOME-APP
```

cf disable-ssh disables SSH access to all instances of an app:

```
$ cf disable-ssh MY-AWESOME-APP
```

Configuring SSH Access at the Space Level

cf allow-space-ssh allows SSH access into all apps in a space:

```
$ cf allow-space-ssh SPACE-NAME
```

cf disallow-space-ssh disallows SSH access into all apps in a space:

```
$ cf disallow-space-ssh SPACE-NAME
```

Checking SSH Permissions

cf ssh-enabled checks whether an app is accessible with SSH:

```
$ cf ssh-enabled MY-AWESOME-APP
ssh support is disabled for 'MY-AWESOME-APP'
```

cf space-ssh-allowed checks whether all apps running within a space are accessible with SSH:

```
$ cf space-ssh-allowed SPACE-NAME
ssh support is enabled in space 'SPACE-NAME'
```

Logging Into an Application Container with cf SSH

If SSH access is allowed at the deployment, space, and application level, you can run the `cf ssh APP-NAME` command to start an interactive SSH session with a VM hosting an application. By default, the command accesses the container running the first instance of the application, the instance with index **0**.

```
$ cf ssh MY-AWESOME-APP
```

When logged into a VM hosting an app, you can use tools like the Cloud Foundry Diego Operator Toolkit (cfdot) to run app status diagnostics. For more information, see How to use CF Diego Operator Toolkit.

Common cf SSH Flags

You can tailor cf ssh commands with the following flags, most of which mimic flags for the Unix or Linux `ssh` command. Run the `cf ssh --help` command for more details.

- The `-i` flag targets a specific instance of an application. To log into the VM container hosting the third instance, `index=2`, of MY-AWESOME-APP, run:

  ```
  $ cf ssh MY-AWESOME-APP -i 2
  ```

- The `-L` flag enables local port forwarding, binding an output port on your machine to an input port on the application VM. Pass in a local port, and your application VM port and port number, all colon delimited. You can prepend your local network interface, or use the default `localhost`.

  ```
  $ cf ssh MY-AWESOME-APP -L [LOCAL-NETWORK-INTERFACE:]LOCAL-PORT:REMOTE-HOST-NAME:REMOTE-HOST-PORT
  ```

- The `-N` flag skips returning a command prompt on the remote machine. This sets up local port forwarding if you do not need to execute commands on the host VM.
- The `--request-pseudo-tty` and `--force-pseudo-tty` flags let you run an SSH session in pseudo-tty mode rather than generate terminal line output.

SSH Session Environment

If you want the environment of your interactive SSH session to match the environment of your buildpack-based app, with the same environment variables and working directory, run the following commands after starting the session:

```
export HOME=/home/vcap/app
export TMPDIR=/home/vcap/tmp
cd /home/vcap/app
```

Before running commands below, verify that the contents of the files in both the `/home/vcap/app/.profile` and `/home/vcap/app/.profile.d` directories will not perform any actions that are undesirable for your running app. The `.profile.d` directory contains buildpack-specific initialization tasks, and the `.profile` file contains application-specific initialization tasks.

If the `profile` and `.profile.d` scripts would alter your instance in undesirable ways, only run the commands in them that you need for environmental setup.

```
[ -d /home/vcap/app/.profile.d ] && for f in /home/vcap/app/.profile.d/*.sh; do source "$f"; done
source /home/vcap/app/.profile
```

After running the above commands, the value of the `VCAP_APPLICATION` environment variable differs slightly from its value in the environment of the app process, as it will not have the `host`, `instance_id`, `instance_index`, or `port` fields set. These fields are available in other environment variables, as described in the VCAP_APPLICATION documentation.

Application SSH Access without cf CLI

In addition to `cf ssh`, you can use other SSH clients such as `ssh`, `scp`, or `sftp` to access your application, if you have SSH permissions.

Follow the steps below to securely connect to an application instance by logging in with a specially-formed username that passes information to the SSH proxy running on the host VM. For the password, use a one-time SSH authorization code generated by cf ssh-code.

1. Run `cf app MY-AWESOME-APP --guid` and record the GUID of your target app.

```
$ cf app MY-AWESOME-APP --guid
abcdefab-1234-5678-abcd-1234abcd1234
```

2. Query the `/v2/info` endpoint of the Cloud Controller in your deployment. Record the domain name and port of the `app_ssh_endpoint` field, and the `app_ssh_host_key_fingerprint` field. You will compare the `app_ssh_host_key_fingerprint` with the fingerprint returned by the SSH proxy on your target VM.

```
$ cf curl /v2/info
{
...
"app_ssh_endpoint": "ssh.MY-DOMAIN.com:2222",
"app_ssh_host_key_fingerprint": "a6:14:c0:ea:42:07:b2:f7:53:2c:0b:60:e0:00:21:6c",
...
}
```

3. Run `cf ssh-code` to obtain a one-time authorization code that substitutes for an SSH password. You can run `cf ssh-code | pbcopy` to automatically copy the code to the clipboard.

```
$ cf ssh-code
E1x89n
```

4. Run your `ssh` or other command to connect to the application instance. For the username, use a string of the form `cf:APP-GUID/APP-INSTANCE-INDEX@SSH-ENDPOINT`, where `APP-GUID` and `SSH-ENDPOINT` come from the previous steps. For the port number, use the `SSH-PORT` recorded above. `APP-INSTANCE-INDEX` is the index of the instance you want to access.

   With the above example, you `ssh` into the container hosting the first instance of your app by running the following command:

```
$ ssh -p 2222 cf:abcdefab-1234-5678-abcd-1234abcd1234/0@ssh.MY-DOMAIN.com
```

   Or you can use `scp` to transfer files by running the following command:

```
$ scp -P 2222 -o User=cf:abcdefab-1234-5678-abcd-1234abcd1234/0 ssh.MY-DOMAIN.com:REMOTE-FILE-TO-RETRIEVE LOCAL-FILE-DESTINATION
```

5. When the SSH proxy reports its RSA fingerprint, confirm that it matches the `app_ssh_host_key_fingerprint` recorded above. When prompted for a password, paste in the authorization code returned by `cf ssh-code`.

```
$ ssh -p 2222 cf:abcdefab-1234-5678-abcd-1234abcd1234/0@ssh.MY-DOMAIN.com
The authenticity of host '[ssh.MY-DOMAIN.com]:2222 ([203.0.113.5]:2222)' can't be established.
RSA key fingerprint is a6:14:c0:ea:42:07:b2:f7:53:2c:0b:60:e0:00:21:6c.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[ssh.MY-DOMAIN.com]:2222 [203.0.113.5]:2222' (RSA) to the list of known hosts.
cf:d0a2e11d-e6ca-4120-b32d-140@ssh.ketchup.cf-app.com's password:
vcap@ce4l5164kws:~$
```

   You have now securely connected to the application instance.

Proxy to Container Authentication

A second layer of SSH security runs within each container. When the SSH proxy attempts to handshake with the SSH daemon inside the target container, it uses the following fields associated with the `diego-ssh` key in its route to the application instance. This inner layer works invisibly and requires no user action, but is described here to complete the SSH security picture.

CONTAINER_PORT (required)

`container_port` indicates which port inside the container the SSH daemon is listening on. The proxy attempts to connect to host side mapping of this port after authenticating the client.

HOST_FINGERPRINT (optional)

When present, `host_fingerprint` declares the expected fingerprint of the SSH daemon's host public key. When the fingerprint of the actual target's host key does not match the expected fingerprint, the connection is terminated. The fingerprint should only contain the hex string generated by `ssh-keygen -l`.

USER (optional)

`user` declares the user ID to use during authentication with the container's SSH daemon. While this is not a required part of the routing data, it is required for password authentication and may be required for public key authentication.

PASSWORD (optional)

`password` declares the password to use during password authentication with the container's ssh daemon.

PRIVATE_KEY (optional)

`private_key` declares the private key to use when authenticating with the container's SSH daemon. If present, the key must be a PEM encoded RSA or DSA public key.

Example Application Process

```
{
  "process_guid": "ssh-process-guid",
  "domain": "ssh-experiments",
  "rootfs": "preloaded:cflinuxfs2",
  "instances": 1,
  "start_timeout": 30,
  "setup": {
    "download": {
      "artifact": "diego-sshd",
      "from": "http://file-server.service.cf.internal.example.com:8080/v1/static/diego-sshd/diego-sshd.tgz",
      "to": "/tmp",
      "cache_key": "diego-sshd"
    }
  },
  "action": {
    "run": {
      "path": "/tmp/diego-sshd",
      "args": [
          "-address=0.0.0.0:2222",
          "-authorizedKey=ssh-rsa ..."
      ],
      "env": [],
      "resource_limits": {}
    }
  },
  "ports": [ 2222 ],
  "routes": {
    "diego-ssh": {
      "container_port": 2222,
      "private_key": "PEM encoded PKCS#1 private key"
    }
  }
}
```

Daemon discovery

To be accessible via the SSH proxy, containers must host an SSH daemon, expose it via a mapped port, and advertise the port in a `diego-ssh` route. If a proxy cannot find the target process or a route, user authentication fails.

```
"routes": {
  "diego-ssh": { "container_port": 2222 }
}
```

The Diego system generates the appropriate process definitions for Pivotal Application Service applications which reflect the policies that are in effect.

Accessing Services with SSH

Page last updated:

This page assumes you are using Cloud Foundry Command Line Interface (cf CLI) v6.15.0 or later.

This topic describes how to gain direct command line access to your deployed service instance. For example, you may need access to your database to execute raw SQL commands to edit the schema, import and export data, or debug application data issues.

To establish direct command line access to a service, you deploy a host app and use its SSH and port forwarding features to communicate with the service instance through the app container. The technique outlined below works with TCP services such as MySQL or Redis.

> **Note**: The procedure in this topic requires use of a service key, and not all services support service keys. Some services support credentials through application binding only.

Create a Service Instance

1. In your terminal window, log in to your deployment with `cf login`.

2. List the marketplace services installed as product tiles on your Pivotal Cloud Foundry (PCF) Ops Manager. See the Adding and Deleting Products topic if you need to add the service as a tile. In this example, we create a p-mysql service instance.

```
$ cf marketplace
p-mysql  100mb MySQL databases on demand
```

3. Create your service instance. As part of the create-service command, indicate the service name, the service plan, and the name you choose for your service instance.

```
$ cf create-service p-mysql 100mb MY-DB
```

Push Your Host App

To push an app that will act as the host for the SSH tunnel, push any app that will successfully deploy to Pivotal Application Service.

> **Note**: Your app must be prepared before you push it. See the Deploy an Application topic for details on preparing apps for deployment.

1. Push your app.

```
$ cf push YOUR-HOST-APP
```

2. Enable SSH for your app.

```
$ cf enable-ssh YOUR-HOST-APP
```

> **Note**: To enable SSH access to your app, SSH access must also be enabled for both the space that contains the app and Pivotal Application Service. See the Application SSH Overview topic for more details.

Create Your Service Key

To establish SSH access to your service instance, you must create a service key that contains critical information for configuring your SSH tunnel.

1. Create a service key for your service instance using the cf create-service-key command.

```
$ cf create-service-key MY-DB EXTERNAL-ACCESS-KEY
```

2. Retrieve your new service key using the cf service-key command.

```
$ cf service-key MY-DB EXTERNAL-ACCESS-KEY
Getting key EXTERNAL-ACCESS-KEY for service instance MY-DB as user@example.com

{
"hostname": "us-cdbr-iron-east-01.p-mysql.net",
"jdbcUrl": "jdbc:mysql://us-cdbr-iron-east-03.p-mysql.net/ad_b2fca6t49704585d?user=b5136e448be920\u0026password=231f435o05",
"name": "ad_b2fca6t49704585d",
"password": "231f435o05",
"port": "3306",
"uri": "mysql://b5136e448be920:231f435o05@us-cdbr-iron-east-03.p-mysql.net:3306/ad_b2fca6t49704585d?reconnect=true",
"username": "b5136e448be920"
}
```

Configure Your SSH Tunnel

Configure an SSH tunnel to your service instance using cf ssh. Tailor the example command below with information from your service key.

```
$ cf ssh -L 63306:us-cdbr-iron-east-01.p-mysql.net:3306 YOUR-HOST-APP
```

- Use any available local port for port forwarding. For example, `63306`.
- Replace `us-cdbr-iron-east-01.p-mysql.net` with the address provided under `hostname` in the service key retrieved above.
- Replace `3306` with the port provided under `port` above.
- Replace `YOUR-HOST-APP` with the name of your host app.

After you enter the command, open another terminal window and perform the steps below in Access Your Service Instance.

Access Your Service Instance

To establish direct command-line access to your service instance, use the relevant command line tool for that service. This example uses the MySQL command line client to access the p-mysql service instance.

```
$ mysql -u b5136e448be920 -h 0 -p -D ad_b2fca6t49704585d -P 63306
```

- Replace `b5136e448be920` with the username provided under `username` in your service key.
- `-h 0` instructs `mysql` to connect to your local machine.
- `-p` instructs `mysql` to prompt for a password. When prompted, use the password provided under `password` in your service key.
- Replace `ad_b2fca6t49704585d` with the database name provided under `name` in your service key.
- `-P 63306` instructs `mysql` to connect on port `63306`.

Trusted System Certificates

Page last updated:

A Cloud Foundry Administrator can deploy a set of trusted system certificates. These trusted certificates are available in Linux-based application instances running on the Diego backend. Such instances include buildpack-based apps using the cflinuxfs2 stack and Docker-image-based apps.

If the administrator configures these certificates, they are available inside the instance containers as files with extension `.crt` in the read-only `/etc/cf-system-certificates` directory.

For cflinuxfs2-based apps, these certificates are also installed directly in the `/etc/ssl/certs` directory, and are available automatically to libraries such as `openssl` that respect that trust store. If the administrator configure these certificates, the location of the certificates is provided in the environment variable `CF_SYSTEM_CERT_PATH` on the instance container.

Cloud Controller API Client Libraries

This topic describes the client libraries available for developers who want to consume the Cloud Controller API (CAPI).

Overview

CAPI is the entry point for most operations within the Cloud Foundry (CF) platform. You can use it to manage orgs, spaces, and apps, which includes user roles and permissions. You can also use CAPI to manage the services provided by your CF deployment, including provisioning, creating, and binding them to apps.

For more information, see the CAPI documentation.

Client Libraries

While you can develop apps that consume CAPI by calling it directly as in the API documentation, you may want to use an existing client library. See the available client libraries below.

Supported

CF currently supports the following clients for CAPI:

- Java
- Scripting with the Cloud Foundry Command Line Interface (cf CLI)

Experimental

The following client is experimental and a work in progress:

- Golang

Unofficial

CF does not support the following clients, but they may be supported by third-parties:

- Golang
- Golang
- Node.js

Using Experimental cf CLI Commands

Page last updated:

This topic describes how to use the experimental Cloud Foundry Command Line Interface (cf CLI) commands offered by the Cloud Controller V3 API.

These commands provide developers with the ability to better orchestrate app deployment workflows. New features include the deployment and management of apps with multiple processes, staging apps with multiple buildpacks, and uploading and staging multiple versions of a single app.

The experimental commands described in this topic require Pivotal Cloud Foundry (PCF) 1.12+ and the cf CLI v6.30.0+.

For more information about cf CLI commands, see the Cloud Foundry CLI Reference Guide. For more information about the Cloud Controller V3 API, see the API documentation.

Note: Because these commands are experimental, they are not guaranteed to be available or compatible in subsequent cf CLI releases.

Overview

The new commands include a `v3-` prefix. While the syntax of some experimental commands is based on the existing cf CLI, these commands call the V3 API and support new flags to unlock additional features. Other commands expose the new primitives of apps, such as by performing operations on an app's packages and droplets.

In the V2 APIs, running and staging an app are tightly coupled operations. As a result, an app cannot be staging and running at the same time. The V3 APIs offer developers more granular control over the uploading, staging, and running of an app.

Commands

Consult the following table for a description of the experimental commands.

Command	Description
`v3-app`	Retrieves and display an app's GUID, suppressing all other health and status output
`v3-apps`	Lists all apps in the target space
`v3-create-app`	Creates a V3 app
`v3-create-package`	Uploads a V3 package
`v3-delete`	Deletes a V3 app
`v3-droplets`	Lists droplets of an app
`v3-env`	Shows all environment variables for an app
`v3-get-health-check`	Shows the type of health check performed on an app
`v3-packages`	Lists packages of an app
`v3-push`	Pushes a new app or syncs changes to an existing app
`v3-restart`	Stops all instances of an app and then starts them again, which may cause downtime
`v3-restart-app-instance`	Terminates and then instantiates an app instance
`v3-set-droplet`	Sets the droplet used to run an app
`v3-set-env`	Sets an environment variable for an app
`v3-set-health-check`	Changes type of health check performed on an app's process
`v3-stage`	Creates a new droplet for an app
`v3-start`	Starts an app
`v3-stop`	Stops an app
`v3-unset-env`	Removes an environment variable from an app

Considerations

Keep in mind the following considerations when using the experimental commands:

- `v3-push` supports only a subset of features of `push`. In particular, it does not support app manifests.
- For some commands, such as `set-env`, `ssh`, and `bind-service`, no new V3 version exists. In those cases, use the old commands.
- You can use V3 and old commands together, but some combinations may give unexpected results. For example, if you use V3 commands to create an app with a package but it is not staged, or you use `v3-push` to push an app but it fails to stage, the old `apps` command does not return the app.

Declaring Multiple App Processes with a Procfile

Developers can supply a Procfile with their app in order to run multiple long-lived processes using a single codebase.

To use a Procfile, include it in the root of your application directory and push your application.

For more information about Procfiles, see the About Procfiles section of the *Production Server Configuration* topic.

Delivering Service Credentials to an Application

Page last updated:

This topic describes binding applications to service instances for the purpose of generating credentials and delivering them to applications. For an overview of services, and documentation on other service management operations, see Using Services. If you are interested in building services for Cloud Foundry and making them available to end users, see the Custom Services documentation.

Bind a Service Instance

Binding a service instance to your application triggers credentials to be provisioned for the service instance and delivered to the application runtime in the VCAP_SERVICES environment variable. For details on consuming these credentials with your application, see Using Bound Service Instances.

Not all services support binding, as some services deliver value to users directly without integration with an application. In many cases binding credentials are unique to an application, and another app bound to the same service instance would receive different credentials; however this depends on the service.

```
$ cf bind-service my-app mydb
Binding service mydb to my-app in org my-org / space test as me@example.com...
OK
TIP: Use 'cf push' to ensure your env variable changes take effect

$ cf restart my-app
```

Note: You must restart or in some cases re-push your application for changes to be applied to the VCAP_SERVICES environment variable and for the application to recognize these changes.

Arbitrary Parameters

Arbitrary parameters require Cloud Foundry Command Line Interface (cf CLI)cf CLI v6.12.1 or later.

Some services support additional configuration parameters with the bind request. These parameters are passed in a valid JSON object containing service-specific configuration parameters, provided either in-line or in a file. For a list of supported configuration parameters, see documentation for the particular service offering.

```
$ cf bind-service rails-sample my-db -c '{"role":"read-only"}'

Binding service my-db to app rails-sample in org console / space development as user@example.com...
OK
```

```
$ cf bind-service rails-sample my-db -c /tmp/config.json

Binding service my-db to app rails-sample in org console / space development as user@example.com... OK
```

Binding with Application Manifest

As an alternative to binding a service instance after pushing an application, you can use the application manifest to bind the service instance during push. As of cf CLI v6.12.1, Arbitrary Parameters are not supported in application manifests.

The following excerpt from an application manifest would bind a service instance called `test-mysql-01` to the application on push.

```
services:
 - test-mysql-01
```

The following excerpt from the `cf push` command and response demonstrates that the cf CLI reads the manifest and binds the service instance to an app called `test-msg-app`.

```
$ cf push
Using manifest file /Users/Bob/test-apps/test-msg-app/manifest.yml

...

Binding service test-mysql-01 to test-msg-app in org My-Org / space development as Bob@shared-domain.example.com
OK
```

For more information about application manifests, see Deploying with Application Manifests.

Using Bound Service Instances

Once you have a service instance created and bound to your application, you need to configure the application to dynamically fetch the credentials for your service instance. The VCAP_SERVICES environment variable contains credentials and additional metadata for all bound service instances. There are two methods developers can leverage to have their applications consume binding credentials.

- **Parse the JSON yourself:** See the documentation for VCAP_SERVICES. Helper libraries are available for some frameworks.
- **Auto-configuration:** Some buildpacks create a service connection for you by creating additional environment variables, updating config files, or passing system parameters to the JVM.

For details on consuming credentials specific to your development framework, refer to the Service Binding section in the documentation for your framework's buildpack.

Update Service Credentials

To update your service credentials, perform the following steps:

1. Unbind the service instance using the credentials you are updating with the following command:

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

2. Bind the service instance with the following command. This adds your credentials to the VCAP_SERVICES environment variable.

   ```
   $ cf bind-service YOUR-APP YOUR-SERVICE-INSTANCE
   ```

3. Restart or re-push the application bound to the service instance so that the application recognizes your environment variable updates.

Unbind a Service Instance

Unbinding a service removes the credentials created for your application from the VCAP_SERVICES environment variable.

```
$ cf unbind-service my-app mydb
Unbinding app my-app from service mydb in org my-org / space test as me@example.com...
OK
```

Note: You must restart or in some cases re-push your application for changes to be applied to the VCAP_SERVICES environment variable and for the application to recognize these changes.

Pivotal

Managing Service Instances with the cf CLI

Page last updated:

This topic describes lifecycle operations for service instances, including creating, updating, and deleting. For an overview of services, and documentation about other service management operations, see the Using Services topic. If you are interested in building services for Cloud Foundry and making them available to end users, see the Custom Services documentation.

To run the commands in this topic, you must first install the Cloud Foundry Command Line Interface (cf CLI). See the Cloud Foundry Command Line Interface topics for more information.

List Marketplace Services

After targeting and logging into Cloud Foundry, run the `cf marketplace` cf CLI command to view the services available to your targeted organization. Available services may differ between organizations and between Cloud Foundry marketplaces.

```
$ cf marketplace
Getting services from marketplace in org my-org / space test as user@example.com...
OK

service       plans          description
p-mysql       100mb, 1gb     A DBaaS
p-riakcs      developer      An S3-compatible object store
```

Creating Service Instances

You can create a service instance with the following command: `cf create-service SERVICE PLAN SERVICE_INSTANCE`

Use the information in the list below to replace `SERVICE`, `PLAN`, and `SERVICE_INSTANCE` with appropriate values.

- `SERVICE`: The name of the service you want to create an instance of.
- `PLAN`: The name of a plan that meets your needs. Service providers use **plans** to offer varying levels of resources or features for the same service.
- `SERVICE_INSTANCE`: The name you provide for your service instance. You use this name to refer to your service instance with other commands. Service instance names can include alpha-numeric characters, hyphens, and underscores, and you can rename the service instance at any time.

```
$ cf create-service rabbitmq small-plan my-rabbitmq

Creating service my-rabbitmq in org console / space development as user@example.com...
OK
```

User Provided Service Instances provide a way for developers to bind applications with services that are not available in their Cloud Foundry marketplace. For more information, see the User Provided Service Instances topic.

Arbitrary Parameters

Arbitrary parameters require cf CLI v6.12.1+

Some services support providing additional configuration parameters with the provision request. Pass these parameters in a valid JSON object containing service-specific configuration parameters, provided either in-line or in a file. For a list of supported configuration parameters, see the documentation for the particular service offering.

Example providing service-specific configuration parameters in-line:

```
$ cf create-service my-db-service small-plan my-db -c '{"storage_gb":4}'

Creating service my-db in org console / space development as user@example.com...
OK
```

Example providing service-specific configuration parameters in a file:

```
$ cf create-service my-db-service small-plan my-db -c /tmp/config.json

Creating service my-db in org console / space development as user@example.com...
OK
```

Instance Tags

Instance tags require cf CLI v6.12.1+

Some services provide a list of tags that Cloud Foundry delivers in the VCAP_SERVICES Environment Variable. These tags provide developers with a more generic way for applications to parse `VCAP_SERVICES` for credentials. Developers may provide their own tags when creating a service instance by including the `-t` flag followed by a comma-separated list of tags.

Example providing a comma-separated list of tags:

```
$ cf create-service my-db-service small-plan my-db -t "prod, workers"

Creating service my-db in org console / space development as user@example.com...
OK
```

List Service Instances

Run the `cf services` command to list the service instances in your targeted space. The output from running this command includes any bound apps and the state of the last requested operation for the service instance.

```
$ cf services
Getting services in org my-org / space test as user@example.com...
OK

name      service     plan       bound apps     last operation
mybucket  p-riakcs    developer  myapp          create succeeded
mydb      p-mysql     100mb                     create succeeded
```

Get Details for a Particular Service Instance

Details include dashboard urls, if applicable, and operation start and last updated timestamps.

```
$ cf service mydb

Service instance: mydb
Service: p-mysql
Plan: 100mb
Description: MySQL databases on demand
Documentation url:
Dashboard: https://p-mysql.example.com/manage/instances/abcd-ef12-3456

Last Operation
Status: create succeeded
Message:
Started: 2015-05-08T22:59:07Z
Updated: 2015-05-18T22:01:26Z
```

Bind a Service Instance

Depending on the service, you can bind service instances to applications and/or routes.

Not all services support binding, as some services deliver value to users directly without integration with Cloud Foundry, such as SaaS applications.

Bind a Service Instance to an Application

Depending on the service, binding a service instance to your application may deliver credentials for the service instance to the application. See the Delivering Service Credentials to an Application topic for more information. Binding a service instance to an application may also trigger application logs to be streamed to the service instance. For more information, see Streaming Application Logs to Log Management Services.

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

$ cf restart my-app
```

Note: You must restart or in some cases re-push your application for changes to be applied to the VCAP_SERVICES environment variable and for the application to recognize these changes.

Binding with Application Manifest

As an alternative to binding a service instance to an application after pushing an application, you can use the application manifest to bind the service instance during push. As of cf CLI v6.12.1, Arbitrary Parameters are not supported in application manifests. Using the manifest to bind service instances to routes is also not supported.

The following excerpt from an application manifest binds a service instance called `test-mysql-01` to the application on push.

```
services:
 - test-mysql-01
```

The following excerpt from the `cf push` command and response demonstrates that the cf CLI reads the manifest and binds the service instance to an app called `test-msg-app`.

```
$ cf push
Using manifest file /Users/Bob/test-apps/test-msg-app/manifest.yml

...

Binding service test-mysql-01 to test-msg-app in org My-Org / space development as user@example.com
OK
```

For more information about application manifests, see Deploying with Application Manifests.

Bind a Service Instance to a Route

Binding a service instance to a route will cause application requests and responses to be proxied through the service instance, where it may be used to transform or intermediate requests. For more information, see Manage Application Requests with Route Services.

```
$ cf bind-route-service shared-domain.example.com --hostname my-app my-service-instance
Binding route my-app.shared-domain.example.com to service instance my-service-instance in org my-org / space test as user@example.com...
OK
```

Restaging your application is not required.

Arbitrary Parameters

Arbitrary parameters require cf CLI v6.12.1+

Some services support additional configuration parameters with the bind request. These parameters are passed in a valid JSON object containing service-specific configuration parameters, provided either in-line or in a file. For a list of supported configuration parameters, see documentation for the particular service offering.

```
$ cf bind-service rails-sample my-db -c '{"role":"read-only"}'

Binding service my-db to app rails-sample in org console / space development as user@example.com...
OK
```

```
$ cf bind-service rails-sample my-db -c /tmp/config.json

Binding service my-db to app rails-sample in org console / space development as user@example.com... OK
```

Unbind a Service Instance

Unbind a Service Instance from an Application

Unbinding a service instance from an application removes the credentials created for your application from the VCAP_SERVICES environment variable.

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

Note: You must restart or in some cases re-push your application for changes to be applied to the VCAP_SERVICES environment variable and for the application to recognize these changes.

Unbind a Service Instance from a Route

Unbinding a service instance from a route will result in requests and responses no longer being proxied through the service instance. For more information, see Manage Application Requests with Route Services.

Note: If your bound service instance is providing security features, like authorization, unbinding the service instance may leave your application vulnerable.

```
$ cf unbind-route-service shared-domain.example.com --hostname my-app my-service-instance

Unbinding may leave apps mapped to route my-app.shared-domain.example.com vulnerable; e.g. if service instance my-service-instance provides authentication. Do you want to proceed?> y

Unbinding route my-app.shared-domain.example.com from service instance my-service-instance n org my-org / space test as user@example.com...
OK
```

Restaging your application is not required.

Rename a Service Instance

You can change the name given to a service instance. Keep in mind that upon restarting any bound applications, the name of the instance will change in the VCAP_SERVICES environment variable. If your application depends on the instance name for discovering credentials, changing the name could break your application's use of the service instance.

```
$ cf rename-service mydb mydb1
Renaming service mydb to mydb1 in org my-org / space test as user@example.com...
OK
```

Update a Service Instance

Upgrade/Downgrade Service Plan

Changing a plan requires cf CLI v6.7+ and cf-release v192+

By updating the service plan for an instance, users can effectively upgrade and downgrade their service instance to other service plans. Though the platform and CLI now support this feature, services must expressly implement support for it so not all services will. Further, a service might support updating between some plans but not others. For instance, a service might support updating a plan where only a logical change is required, but not where data migration is necessary. In either case, users can expect to see a meaningful error when plan update is not supported.

```
$ cf update-service mydb -p new-plan
Updating service instance mydb as user@example.com...
OK
```

Arbitrary Parameters

Arbitrary parameters require cf CLI v6.12.1+

Some services support additional configuration parameters with the update request. These parameters are passed in a valid JSON object containing service-specific configuration parameters, provided either in-line or in a file. For a list of supported configuration parameters, see documentation for the particular service offering.

```
$ cf update-service mydb -c '{"storage_gb":4}'

Updating service instance mydb as me@example.com...
```

```
$ cf update-service mydb -c /tmp/config.json

Updating service instance mydb as user@example.com...
```

Instance Tags

Instance tags require cf CLI v6.12.1+

Some services provide a list of tags that Cloud Foundry delivers in the VCAP_SERVICES Environment Variable. These tags provide developers with a more generic way for applications to parse `VCAP_SERVICES` for credentials. Developers may provide their own tags when creating a service instance by including a comma-separated list of tags with the `-t` flag.

```
$ cf update-service my-db -t "staging, web"

Updating service my-db in org console / space development as user@example.com...
OK
```

Delete a Service Instance

Deleting a service instance deprovisions the service instance and deletes all data associated with the service instance.

```
$ cf delete-service mydb

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

Managing Service Keys

Page last updated:

This topic describes managing service instance credentials with service keys.

Service keys generate credentials for manually configuring consumers of marketplace services. Once you configure them for your service, local clients, apps in other spaces, or entities outside your deployment can access your service with these keys.

> **Note**: Some service brokers do not support service keys. If you want to build a service broker that supports service keys, see Services. If you want to use a service broker that does not support service keys, see Delivering Service Credentials to an Application.

Create a Service Key

To generate credentials for a service instance, use the `cf create-service-key` command:

```
$ cf create-service-key MY-SERVICE MY-KEY
Creating service key MY-KEY for service instance MY-SERVICE as me@example.com...
OK
```

Use the `-c` flag to provide service-specific configuration parameters in a valid JSON object, either in-line or in a file.

To provide the JSON object in-line, use the following format:

```
$ cf create-service-key MY-SERVICE MY-KEY -c '{"permissions":"read-only"}'
Creating service key MY-KEY for service instance MY-SERVICE as me@example.com...
OK
```

To provide the JSON object as a file, give the absolute or relative path to your JSON file:

```
$ cf create-service-key MY-SERVICE MY-KEY -c PATH-TO-JSON-FILE
Creating service key MY-KEY for service instance MY-SERVICE as me@example.com...
OK
```

List Service Keys for a Service Instance

To list service keys for a service instance, use the `cf service-keys` command:

```
$ cf service-keys MY-SERVICE
Getting service keys for service instance MY-SERVICE as me@example.com...

name
mykey1
mykey2
```

Get Credentials for a Service Key

To retrieve credentials for a service key, use the `cf service-key` command:

```
$ cf service-key MY-SERVICE MY-KEY
Getting key MY-KEY for service instance MY-SERVICE as me@example.com...

{
  uri: foo://user2:pass2@example.com/mydb,
  servicename: mydb
}
```

Use the `--guid` flag to display the API GUID for the service key:

```
$ cf service-key --guid MY-SERVICE MY-KEY
Getting key MY-KEY for service instance MY-SERVICE as me@example.com...

e3696fcb-7a8f-437f-8692-436558e45c7b

OK
```

Configure Credentials for a Service Key

Once these credentials are obtained, you can use a local CLI or utility to connect to the service instance, configure an application running outside the platform to connect to the service instance, or create a user-provided service instance so that applications in another space can connect to the service instance. How you configure these credentials will depend on what local client, app, or entity is used to access your service instance.

For more information on configuring credentials with a user-provided service instance, see User-Provided Service Instances.

Delete a Service Key

To delete a service key, use the `cf delete-service-key` command:

```
$ cf delete-service-key MY-SERVICE MY-KEY

Are you sure you want to delete the service key MY-KEY ? y
Deleting service key MY-KEY for service instance MY-SERVICE as me@example.com...

OK
```

Add option `-f` to force deletion without confirmation.

```
$ cf delete-service-key -f MY-SERVICE MY-KEY

Deleting service key MY-KEY for service instance MY-SERVICE as me@example.com...

OK
```

User-Provided Service Instances

Page last updated:

This topic describes how to create and update user-provided service instances.

> **Note**: The procedures in this topic use the Cloud Foundry Command Line Interface (cf CLI). You can also create user-provided service instances in Apps Manager from the Marketplace. To update existing user-provided service instances, navigate to the service instance page and select the **Configuration** tab.

Overview

User-provided service instances enable developers to use services that are not available in the marketplace with their applications running on Cloud Foundry.

User-provided service instances can be used to deliver service credentials to an application, and/or to trigger streaming of application logs to a syslog compatible consumer. These two functions can be used alone or at the same time.

Once created, user-provided service instances behave like service instances created through the marketplace; see Managing Service Instances and Application Binding for details on listing, renaming, deleting, binding, and unbinding.

Create a User-Provided Service Instance

The alias for cf create-user-provided-service is `cf cups`.

Deliver Service Credentials to an Application

Suppose a developer obtains a URL, port, username, and password for communicating with an Oracle database managed outside of Cloud Foundry. The developer could manually create custom environment variables to configure their application with these credentials (of course you would never hard code these credentials in your application!).

User-provided service instances enable developers to configure their applications with these using the familiar Application Binding operation and the same application runtime environment variable used by Cloud Foundry to automatically deliver credentials for marketplace services (VCAP_SERVICES).

```
cf cups SERVICE_INSTANCE -p '{"username":"admin","password":"pa55woRD"}'
```

To create a service instance in interactive mode, use the `-p` option with a comma-separated list of parameter names. The Cloud Foundry Command Line Interface (cf CLI) prompts you for each parameter value.

```
$ cf cups my-user-provided-route-service -p "host, port"

host> rdb.local

port> 5432

Creating user provided service my-user-provided-route-service in org my-org / space my-space as user@example.com...
OK
```

Once the user-provided service instance is created, to deliver the credentials to one or more applications see Application Binding.

Stream Application Logs to a Service

User-provided service instances enable developers to stream applications logs to a syslog compatible aggregation or analytics service that isn't available in the marketplace. For more information about the syslog protocol see RFC 5424 and RFC 6587.

Create the user-provided service instance, specifying the URL of the service with the `-l` option.

```
cf cups SERVICE_INSTANCE -l syslog://example.log-aggregator.com
```

To stream application logs to the service, bind the user-provided service instance to your app.

Proxy Application Requests to a Route Service

User-provided service instances enable developers to proxy application requests to route services for preprocessing. To create a user-provided service instance for a route service, specify the url for the route service using the `-r` option.

```
$ cf create-user-provided-service my-user-provided-route-service -r https://my-route-service.example.com
Creating user provided service my-user-provided-route-service in org my-org / space my-space as user@example.com...
OK
```

Note: When creating the user-provided service, the route service url specified must be https.

To proxy requests to the user-provided route service, you must bind the service instance to the route. For more information, see Manage Application Requests with Route Services.

Update a User-provided Service Instance

You can use cf update-user-provided-service to update the attributes of an instance of a user-provided service. New credentials overwrite old credentials, and parameters not provided are deleted.

The alias for `update-user-provided-service` is `uups`.

Streaming Application Logs to Log Management Services

Page last updated:

This topic describes how to drain logs from Cloud Foundry to a third-party log management service.

Cloud Foundry aggregates logs for all instances of your applications as well as for requests made to your applications through internal components of Cloud Foundry. For example, when the Cloud Foundry Router forwards a request to an application, the Router records that event in the log stream for that app. Run the following command to access the log stream for an app in the terminal:

```
$ cf logs YOUR-APP-NAME
```

If you want to persist more than the limited amount of logging information that Cloud Foundry can buffer, drain these logs to a log management service.

For more information about the systems responsible for log aggregation and streaming in Cloud Foundry, see Application Logging in Cloud Foundry.

Using Services from the Cloud Foundry Marketplace

Your Cloud Foundry marketplace may offer one or more log management services. To use one of these services, create an instance of the service and bind it to your application with the following commands:

```
$ cf create-service SERVICE PLAN SERVICE-INSTANCE
$ cf bind-service YOUR-APP YOUR-LOG-STORE
```

For more information about service instance lifecycle management, see the Managing Service Instances topic.

> **Note**: Not all marketplace services support syslog drains. Some services implement an integration with Cloud Foundry that enables automated streaming of application syslogs. If you are interested in building services for Cloud Foundry and making them available to end users, see the Custom Services documentation.

Using Services Not Available in your Marketplace

If a compatible log management service is not available in your Cloud Foundry marketplace, you can use User-provided Service Instances to stream application logs to a service of your choice.

Your service may require some preparation before application logs can be streamed to it from Cloud Foundry. For specific instructions for several popular services, see Service-Specific Instructions for Streaming Application Logs. If you cannot find instructions for your service, follow the generic instructions below.

Step 1: Configure the Log Management Service

Complete the following steps to set up a communication channel between the log management service and your Cloud Foundry deployment:

1. Obtain the external IP addresses that your Cloud Foundry administrator assigns to outbound traffic.
2. Provide these IP addresses to the log management service. The specific steps to configure a third-party log management service depend on the service.
3. Whitelist these IP addresses to ensure unrestricted log routing to your log management service.
4. Record the syslog URL provided by the third-party service. Third-party services typically provide a syslog URL to use as an endpoint for incoming log data. You use this syslog URL in Step 2: Create a User-provided Service Instance.

   Cloud Foundry uses the syslog URL to route messages to the service. The syslog URL has a scheme of `syslog`, `syslog-tls`, or `https`, and can include a port number. For example:
   `syslog://logs.example.com:1234`

   > **Note**: PAS does not support using `syslog-tls` with self-signed certificates. If you are running your own syslog server and want to use `syslog-`

`tls`, you must have an SSL certificate signed by a well-known certificate authority.

Step 2: Create a User-provided Service Instance

Create a user-provided service instance using the Cloud Foundry Command Line Interface (cf CLI) `create-user-provided-service` command with the `-l` flag and the syslog URL that you obtained in *Step 1: Configure the Log Management Service*. The `-l` flag configures the syslog drain.

```
$ cf create-user-provided-service SERVICE-INSTANCE -l SYSLOG-URL
```

Refer to User-Provided Service Instances for more information.

Step 3: Bind the Service Instance

You have two options for binding the service instance to an application:

- Run `cf push` with a manifest. The services block in the manifest must specify the service instance that you want to bind.
- Run `cf bind-service`

```
$ cf bind-service YOUR-APP-NAME SERVICE-INSTANCE
```

After a short delay, logs begin to flow automatically. Refer to Managing Service Instances with the CLI for more information.

Step 4: Verify Logs are Draining

To verify that logs are draining correctly to a third-party log management service:

1. Take actions that produce log messages, such as making requests of your app.
2. Compare the logs displayed in the CLI against those displayed by the log management service.

For example, if your application serves web pages, you can send HTTP requests to the application. In Cloud Foundry, these generate Router log messages, which you can view in the CLI. Your third-party log management service should display corresponding messages.

Note: For security reasons, Cloud Foundry applications do not respond to `ping`. You cannot use `ping` to generate log entries.

Service-Specific Instructions for Streaming Application Logs

Page last updated:

This topic provides instructions for configuring some third-party log management services.

Once you have configured a service, refer to the Third-Party Log Management Services topic for instructions on binding your application to the service.

Logit.io

From your Logit.io dashboard:

1. Identify the Logit ELK stack you want to use.
2. Click Logstash **Configuration**.
3. Note your Logstash **Endpoint**.
4. Note your TCP-SSL, TCP, or UDP **Port** (not the syslog port).
5. Create the log drain service in Cloud Foundry.

   ```
   $ cf cups logit-ssl-drain -l syslog-tls://ENDPOINT:PORT
   ```

   or

   ```
   $ cf cups logit-drain -l syslog://ENDPOINT:PORT
   ```

6. Bind the service to an app.

   ```
   $ cf bind-service YOUR-CF-APP-NAME logit-ssl-drain
   ```

   or

   ```
   $ cf bind-service YOUR-CF-APP-NAME logit-drain
   ```

7. Restage or push the app using one of the following commands:

   ```
   $ cf restage YOUR-CF-APP-NAME
   ```

   ```
   $ cf push YOUR-CF-APP-NAME
   ```

   After a short delay, logs begin to appear in Kibana.

Papertrail

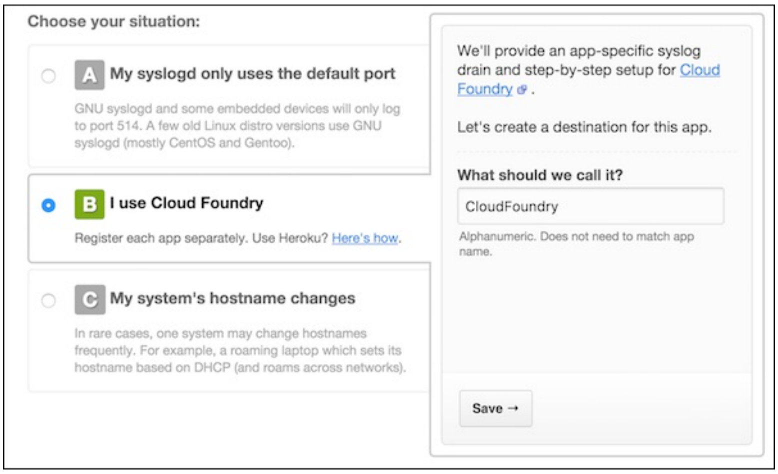
From your Papertrail account:

1. Click **Add System**.

2. Click the **Other** link.

3. Select **I use Cloud Foundry**, enter a name, and click **Save**.



4. Record the URL with port that is displayed after creating the system.

5. Create the log drain service in Cloud Foundry.

```
$ cf cups my-logs -l syslog-tls://logs.papertrailapp.com:PORT
```

6. Bind the service to an app.

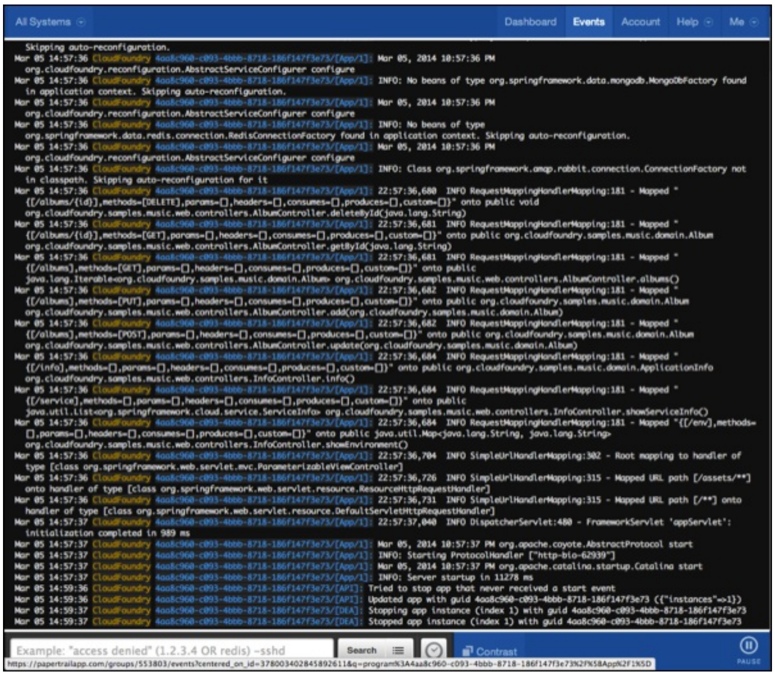
```
$ cf bind-service APPLICATION-NAME my-logs
```

7. Restage the app.

```
$ cf restage APPLICATION-NAME
```

After a short delay, logs begin to flow automatically.

8. Once Papertrail starts receiving log entries, the view automatically updates to the logs viewing page.



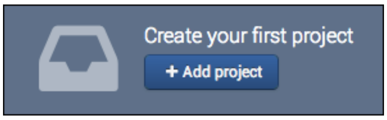
Splunk

See [Streaming Application Logs to Splunk](#) for details.

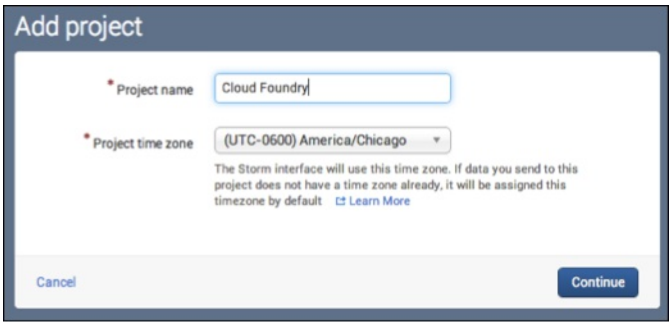
Splunk Storm

From your Splunk Storm account:

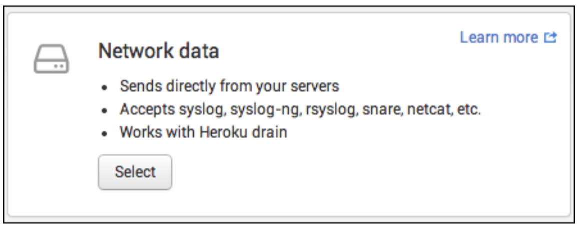
1. Click **Add project**.



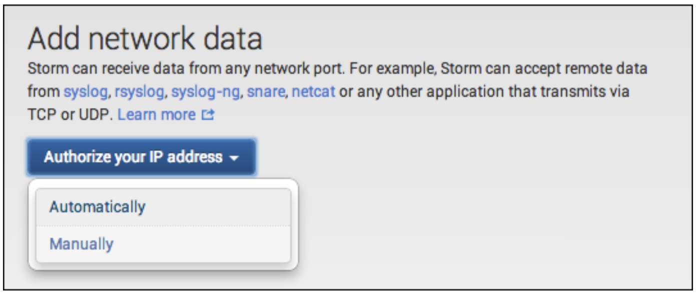
2. Enter the project details.



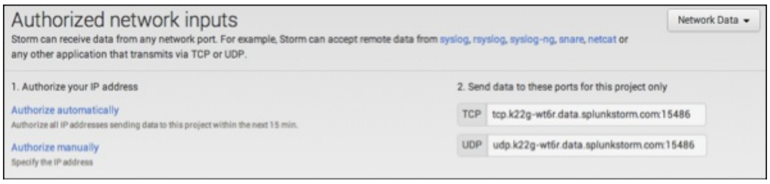
3. Create a new **input** for **Network data**.



4. Manually enter the external IP addresses your Cloud Foundry administrator assigns to outbound traffic.



5. Note the host and port provided for TCP input.



6. Create the log drain service in Cloud Foundry using the displayed TCP host and port.

```
$ cf cups my-logs -l syslog://HOST:PORT
```

7. Bind the service to an app

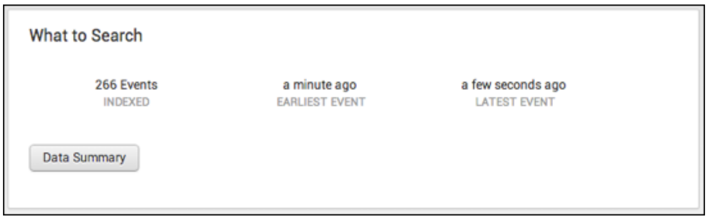
```
$ cf bind-service APPLICATION-NAME my-logs
```

8. Restage the app

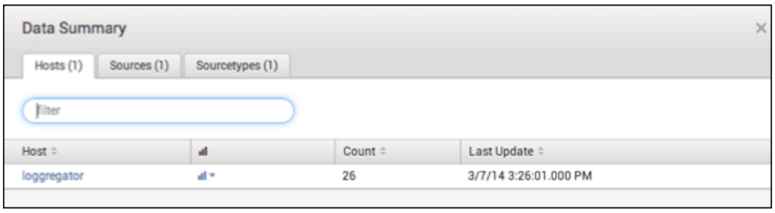
```
$ cf restage APPLICATION-NAME
```

   After a short delay, logs begin to flow automatically.

9. Wait for some events to appear, then click **Data Summary**.



10. Click the **loggregator** link to view all incoming log entries from Cloud Foundry.



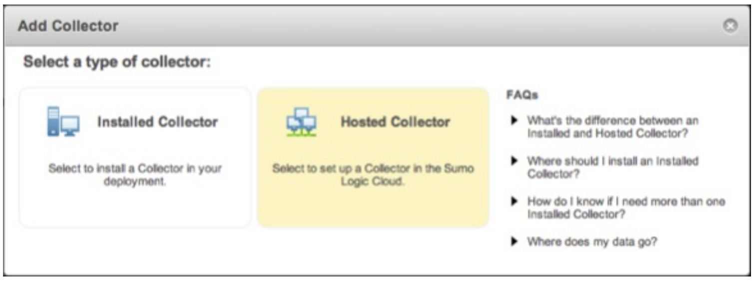
SumoLogic

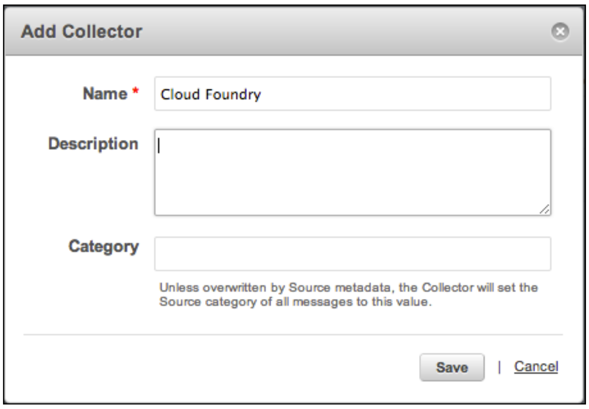
Note: SumoLogic uses HTTPS for communication. HTTPS is supported in Cloud Foundry v158 and later.

From your SumoLogic account:

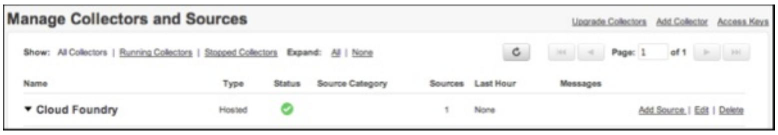
1. Click the **Add Collector** link.

2. Choose **Hosted Collector** and fill in the details.

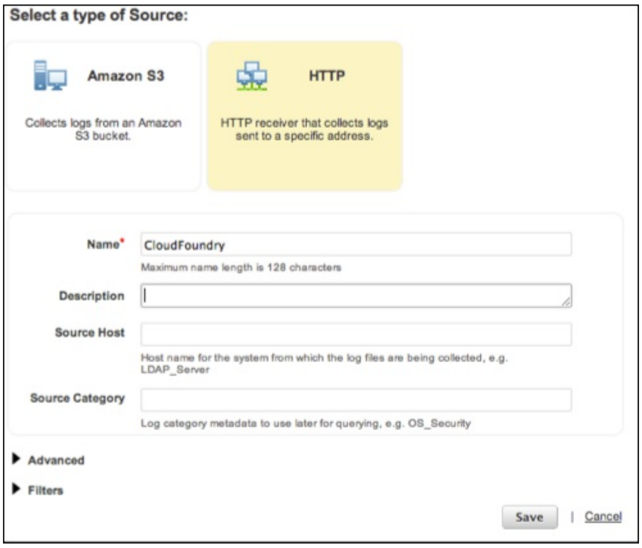




3. In the new collector's row of the collectors view, click the **Add Source** link.



4. Select **HTTP** source and fill in the details. Note that you'll be provided an HTTPS url



5. Once the source is created, a URL should be displayed. You can also view the URL by clicking the **Show URL** link beside the created source.

6. Create the log drain service in Cloud Foundry using the displayed URL.

```
$ cf cups my-logs -l HTTPS-SOURCE-URL
```

7. Bind the service to an app.

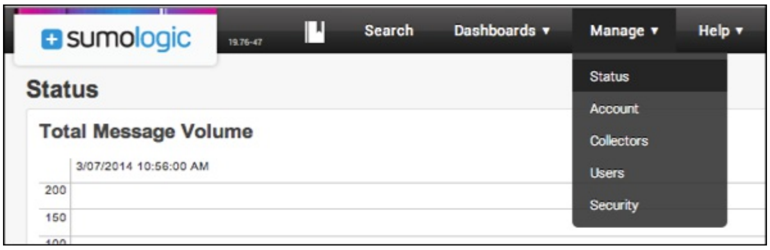
```
$ cf bind-service APPLICATION-NAME my-logs
```

8. Restage the app.

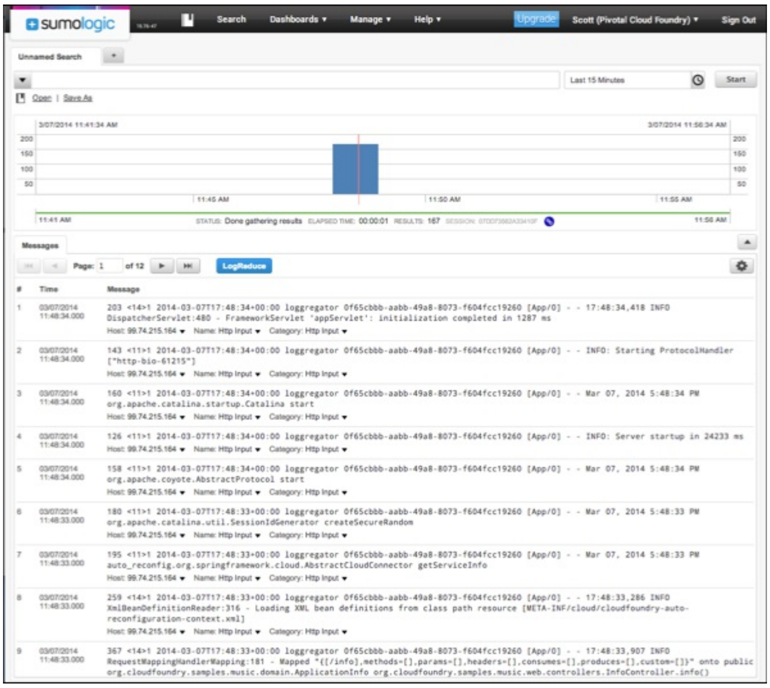
```
$ cf restage APPLICATION-NAME
```

   After a short delay, logs begin to flow automatically.

9. In the SumoLogic dashboard, click **Manage**, then click **Status** to see a view of log messages received over time.



10. In the SumoLogic dashboard, click **Search**. Place the cursor in the search box, then press **Enter** to submit an empty search query.



Logsene

Note: Logsene uses HTTPS for communication. HTTPS is supported in Cloud Foundry v158 and later.

From your Sematext account:

1. Click the Create App / Logsene App menu item. Enter a name and click **Add Application** to create the Logsene App.

2. Create the log drain service in Cloud Foundry using the displayed URL.

```
$ cf cups logsene-log-drain -l https://logsene-cf-receiver.sematext.com/YOUR_LOGSENE_TOKEN
```

3. Bind the log drain to an app. You could optionally bind multiple apps to one log drain.

```
$ cf bind-service YOUR-CF-APP-NAME logsene-log-drain
```

4. Restage the app.

```
$ cf restage APPLICATION-NAME
```

After a short delay, logs begin to flow automatically and appear in the Logsene UI.

Logentries is Not Supported

Cloud Foundry distributes log messages over multiple servers to handle load. Currently, we do not recommend using Logentries as it does not support multiple syslog sources.

Streaming Application Logs to Splunk

Page last updated:

To integrate Cloud Foundry with Splunk Enterprise, complete the following process.

1. Create a Cloud Foundry Syslog Drain for Splunk

In Cloud Foundry, create a syslog drain user-provided service instance as described in Using Third-Party Log Management Services.

Choose one or more applications whose logs you want to drain to Splunk through the service.

Bind each app to the service instance and restart the app.

Note the GUID for each app, the IP address of the Loggregator host, and the port number for the service. Locate the port number in the syslog URL. For example:

`syslog://logs.example.com:1234`

2. Prepare Splunk for Cloud Foundry

For detailed information about the following tasks, see the Splunk documentation.

Install the RFC5424 Syslog Technology Add-On

The Cloud Foundry Loggregator component formats logs according to the Syslog Protocol defined in RFC 5424. Splunk does not parse log fields according to this protocol. To allow Splunk to correctly parse RFC 5424 log fields, install the Splunk RFC5424 Syslog Technical Add-On.

Patch the RFC5424 Syslog Technology Add-On

1. SSH into the Splunk VM
2. Replace `/opt/splunk/etc/apps/rfc5424/default/transforms.conf` with a new `transforms.conf` file that consists of the following text:

```
[rfc5424_host]
DEST_KEY = MetaData:Host
REGEX = <\d+>\d{1}\s{1}\S+\s{1}(\S+)
FORMAT = host::$1

[rfc5424_header]
REGEX = <(\d+)>\d{1}\s{1}\S+\s{1}\S+\s{1}(\S+)\s{1}(\S+)\s{1}(\S+)
FORMAT = prival::$1 appname::$2 procid::$3 msgid::$4
MV_ADD = true
```

3. Restart Splunk

Create a TCP Syslog Data Input

Create a TCP Syslog Data Input in Splunk, with the following settings:

- **TCP port** is the port number you assigned to your log drain service
- **Set sourcetype** is `Manual`
- **Source type** is `rfc5424_syslog` (type this value into text field)
- **Index** is the index you created for your log drain service

Your Cloud Foundry syslog drain service is now integrated with Splunk.

Pivotal

3. Verify that Integration was Successful

Use Splunk to execute a query of the form:

```
sourcetype=rfc5424_syslog index=-THE-INDEX-YOU-CREATED appname=APP-GUID
```

To view logs from all apps at once, you can omit the `appname` field.

Verify that results rows contain the three Cloud Foundry-specific fields:

- **appname**: The GUID for the Cloud Foundry application
- **host**: The IP address of the Loggregator host
- **procid**: The Cloud Foundry component emitting the log

If the Cloud Foundry-specific fields appear in the log search results, integration is successful.

If logs from an app are missing, make sure that the following are true:

- The app is bound to the service and was restarted after binding
- The service port number matches the TCP port number in Splunk

Pivotal

Streaming Application Logs with Fluentd

Page last updated:

Fluentd is an open source log collector that allows you to implement unified logging layers. With Fluentd, you can stream application logs to different backends or services like Elasticsearch, HDFS and Amazon S3. This topic explains how to integrate Fluentd with Cloud Foundry applications.

Step 1: Create a Cloud Foundry Syslog Drain for Fluentd

1. In Cloud Foundry, create a syslog drain user-provided service instance as described in Using Third-Party Log Management Services.
2. Choose one or more applications whose logs you want to drain to Fluentd through the service.
3. Bind each app to the service instance, and restart the app.
4. Note the GUID for each app, the IP address of the Loggregator host, and the port number for the service.
5. Locate the port number in the syslog URL. For example:
   `syslog://logs.example.com:5140`

Step 2: Set up Fluentd for Cloud Foundry

This section assumes you have an active Fluentd instance running. If you do not have an active Fluentd instance, refer to the Fluentd Documentation/Install steps for more details.

Fluentd comes with native support for syslog protocol. To set up Fluentd for Cloud Foundry, configure the syslog input of Fluentd as follows.

1. In your main Fluentd configuration file, add the following `source` entry:

```
<source>
  @type syslog
  port 5140
  bind 0.0.0.0
  tag cf.app
  protocol_type udp
</source>
```

2. Restart the Fluentd service.

Note: The Fluentd syslog input plugin supports `udp` and `tcp` options. Make sure to use the same transport that Cloud Foundry is using.

Fluentd will start listening for Syslog message on port 5140 and tagging the messages with `cf.app`, which can be used later for data routing. For more details about the full setup for the service, refer to the Config File article.

If your goal is to use an Elasticsearch or Amazon S3 backend, read the following guide:http://www.fluentd.org/guides/recipes/elasticsearch-and-s3

Pivotal

Streaming Application Logs to Azure OMS Log Analytics (Beta)

Page last updated:

> **WARNING**: The OMS Log Analytics Firehose Nozzle is currently intended for evaluation and test purposes only. Do not use this product in a production environment.

This topic explains how to integrate your Cloud Foundry (CF) apps with OMS Log Analytics.

Operations Management Suite (OMS) Log Analytics is a monitoring service for Microsoft Azure. The OMS Log Analytics Firehose Nozzle is a CF component that forwards metrics from the Loggregator Firehose to OMS Log Analytics.

This topic assumes you are using the latest version of the Cloud Foundry Command Line Interface (cf CLI) and a working Pivotal Application Service deployment on Azure.

Step 1: Create an OMS Workspace in Azure

See Get started with Log Analytics in the Microsoft Azure documentation to create an OMS workspace.

Step 2: Deploy the Nozzle to Cloud Foundry

1. Run `cf login -a https://api.YOUR-DOMAIN -u YOUR-USERNAME --skip-ssl-validation`, replacing `YOUR-DOMAIN` with your domain and `YOUR-USERNAME` with your CF username, to authenticate to your CF instance. For example:

   ```
   $ cf login -a https://api.example.com -u admin --skip-ssl-validation
   ```

2. Follow the steps below to create a new Cloud Foundry user and grant it access to the Loggregator Firehose using the UAA CLI (UAAC). For more information, see Creating and Managing Users with the UAA CLI (UAAC) and Orgs, Spaces, Roles, and Permissions.

   a. Use `uaac target uaa.YOUR-DOMAIN` to target your UAA server:

   ```
   $ uaac target uaa.example.com --skip-ssl-validation
   ```

   b. Run the following command to obtain an access token for the admin client:

   ```
   $ uaac token client get admin
   ```

   c. Run `cf create-user USERNAME PASSWORD`, replacing `USERNAME` with a new username and `PASSWORD` with a password, to create a new user. For example:

   ```
   $ cf create-user firehose-user firehose-password
   ```

   d. Run `uaac member add cloud_controller.admin USERNAME`, replacing `USERNAME` with the new username, to grant the new user admin permissions. For example:

   ```
   $ uaac member add cloud_controller.admin firehose-user
   ```

   e. Run `uaac member add doppler.firehose USERNAME`, replacing `USERNAME` with the new username, to grant the new user permission to read logs from the Loggregator Firehose endpoint. For example:

   ```
   $ uaac member add doppler.firehose firehose-user
   ```

3. Download the OMS Log Analytics Firehose Nozzle BOSH release from Github. Clone the repository and navigate to the `oms-log-analytics-firehose-nozzle` directory:

   ```
   $ git clone https://github.com/Azure/oms-log-analytics-firehose-nozzle.git
   $ cd oms-log-analytics-firehose-nozzle
   ```

4. Set the following environment variables in the OMS Log Analytics Firehose Nozzle manifest:

Environment Variable	Description

`applications:`
`- name: oms_nozzle`
`...`
`env:`
`  OMS_WORKSPACE: YOUR-WORKSPACE-ID`
`  OMS_KEY: YOUR-OMS-KEY`
	Enter the ID and key value for your OMS workspace.

`OMS_POST_TIMEOUT: 10s`	(Optional) Set the HTTP post timeout for sending events to OMS Log Analytics. The default value is 10 seconds.
`OMS_BATCH_TIME: 10s`	(Optional) Set the interval for posting a batch to OMS. The default value is 10 seconds. For more information, see the Configure Additional Logging section below.
`OMS_MAX_MSG_NUM_PER_BATCH: 1000`	(Optional) Set the maximum number of messages to include in an OMS batch. The default amount is 1000. For more information, see the Configure Additional Logging section below.
`FIREHOSE_USER: YOUR-FIREHOSE-USER` `FIREHOSE_USER_PASSWORD: YOUR-FIREHOSE-PASSWORD`	Enter the username and password for the Firehose user you created in Step 2c.
`API_ADDR: https://api.YOUR-DOMAIN`	Enter the URL of your API endpoint.
`DOPPLER_ADDR: wss://doppler.YOUR-DOMAIN:443`	Enter the URL of your Loggregator traffic controller endpoint.
`EVENT_FILTER: YOUR-LIST`	(Optional) Enter the event types you want to filter out in a comma-separated list. The valid event types are `METRIC`, `LOG`, and `HTTP`.
`IDLE_TIMEOUT: 60s`	(Optional) Set the duration for the Firehose keepalive connection. The default time is 60 seconds.
`SKIP_SSL_VALIDATION: TRUE-OR-FALSE`	Set this value to `TRUE` to allow insecure connections to the UAA and the traffic controller. To block insecure connections to the UAA and traffic controller, set this value to `FALSE`.
`LOG_LEVEL: INFO`	(Optional) Change this value to increase or decrease the amount of logs. Valid log levels in increasing order include `INFO`, `ERROR`, and `DEBUG`. The default value is `INFO`.
`LOG_EVENT_COUNT: TRUE-OR-FALSE`	Set this value to `TRUE` to log the total count of events that the nozzle has sent, received, and lost. OMS logs this value as `CounterEvents`. For more information, see the Configure Additional Logging section below.
`LOG_EVENT_COUNT_INTERVAL: 60s`	(Optional) Set the time interval for logging the event count to OMS. The default interval is 60 seconds. For more information, see the Configure Additional Logging section below.

5. Push the app:

```
$ cf push
```

Step 3: View Logs in OMS Portal

Import the Cloud Foundry OMS view to your OMS Portal to view visualized logs and metrics. You can also create alert rules for specific events.

💡 **Note**: The OMS view of Cloud Foundry is not yet available in the OMS Solutions Gallery. You can add it manually to view your logs in OMS Portal.

Import the OMS View

1. From the main OMS Overview page, navigate to **View Designer**.
2. Click **Import**.
3. Click **Browse**.
4. Select the **Cloud Foundry (Preview).omsview** file.
5. Save the view. The main OMS Overview page displays the **Tile**.
6. Click the **Tile** to view visualized metrics.

See the OMS Log Analytics View Designer documentation for more information.

Create Alert Rules

See Understanding alerts in Log Analytics for more information about OMS Log Analytics alerts.

Set Alert Queries

This section includes example queries that operators can set in the OMS Portal.

- The following query alerts the operator when the nozzle sends a `slowConsumerAlert` to OMS:

```
Type=CF_ValueMetric_CL Name_s=slowConsumerAlert
```

- The following query alerts the operator when Loggregator sends an `LGR` to indicate problems with the logging process:

```
Type=CF_LogMessage_CL SourceType_s=LGR MessageType_s=ERR
```

- The following query alerts the operator when the number of lost events reaches a certain threshold, specified in the OMS Portal:

```
Type=CF_CounterEvent_CL Job_s=nozzle Name_s=eventsLost
```

- The following query alerts the operator when the nozzle receives the `TruncatingBuffer.DroppedMessages` **CounterEvent**:

```
Type=CF_CounterEvent_CL Name_s="TruncatingBuffer.DroppedMessages"
```

(Optional) Step 4: Configure Additional Logging

OMS Log Analytics Firehose Nozzle forwards metrics from the Loggregator Firehose to OMS with minimal processing, but the nozzle can push additional metrics to OMS.

Log Sent, Received, and Lost Events

If you set the `LOG_EVENT_COUNT` environment variable to `TRUE` in the manifest, the nozzle periodically sends the count of sent, received, and lost events to OMS. The value you set for the `LOG_EVENT_COUNT_INTERVAL` determines how frequently the nozzle sends the count.

💡 **Note**: The nozzle does not count **CounterEvents** themselves in the sent, received, or lost event count.

The nozzle sends the count as a **CounterEvent** with a **CounterKey** of one of the following:

CounterEvent	CounterKey
`nozzle.stats.eventsReceived`	The number of events the Firehose has received during the interval
`nozzle.stats.eventsSent`	The number of events the nozzle has successfully sent to OMS during the interval
`nozzle.stats.eventsLost`	The number of events the nozzle has tried to send to OMS during the interval, but failed to send after 4 attempts

In most cases, the total count of `eventsSent` plus `eventsLost` is less than the total `eventsReceived` at the same time. The nozzle buffers some messages and posts them in a batch to OMS. Operators can adjust the buffer size by adjusting the `OMS_BATCH_TIME` and `OMS_MAX_MSG_NUM_PER_BATCH` environment variables in the manifest.

Log Slow Consumer Alerts

Note: The nozzle does not count **ValueMetrics** in the sent, received, or lost event count.

Loggregator sends the nozzle a `slowConsumerAlert` in the following situations:

- WebSocket sends the error code `ClosePolicyViolation (1008)`
- The nozzle receives a **CounterEvent** with the value `TruncatingBuffer.DroppedMessages`

In either case, the nozzle sends the `slowConsumerAlert` event to OMS as the following **ValueMetric**:

ValueMetric	MetricKey
`nozzle.alert.slowConsumerAlert`	1

See the *Slow Nozzle Alerts* section of the Loggregator Guide for Cloud Foundry Operators for more information.

(Optional) Step 5: Scale the Deployment

Scale the Nozzle

If the nozzle is unable to keep up with processing logs from the Firehose, Loggregator alerts the nozzle. When the nozzle receives the alert, it sends a `slowConsumerAlert` to OMS. If this happens, scaling up the nozzle minimizes data loss.

If an operator chooses to scale up their deployment, the Firehose evenly distributes events across all instances of the nozzle. See the *Scaling Nozzles* section of the Loggregator Guide for Cloud Foundry Operators for more information.

Operators can create an alert rule for the `slowConsumerAlert` message.

Scale Loggregator

Loggregator sends `LGR` log messages to indicate problems with the logging process. See the *Scaling Loggregator* section of the Loggregator Guide for Cloud Foundry Operators for more information.

Operators can create an alert rule for the `LGR` message.

Configuring Play Framework Service Connections

Page last updated:

Cloud Foundry provides support for connecting a Play Framework application to services such as MySQL, and Postgres. In many cases, a Play Framework application running on Cloud Foundry can automatically detect and configure connections to services.

Auto-Configuration

By default, Cloud Foundry will detect service connections in a Play Framework application and configure them to use the credentials provided in the Cloud Foundry environment. Auto-configuration will only happen if there is a single service of any of the supported types - MySQL or Postgres.

Migrating a Database in Cloud Foundry

Page last updated:

Application development and maintenance often requires changing a database schema, known as migrating the database. This topic describes three ways to migrate a database on Cloud Foundry.

You can also migrate a database by running a task with the Cloud Foundry Command Line Interface tool (cf CLI). For more information about running tasks in Cloud Foundry, see the Running Tasks topic.

Migrate Once

This method executes SQL commands directly on the database, bypassing Cloud Foundry. This is the fastest option for a single migration. However, this method is less efficient for multiple migrations because it requires manually accessing the database every time.

> **Note:** Use this method if you expect your database migration to take longer than the timeout that `cf push` applies to your application. The timeout defaults to 60 seconds, but you can extend it up to 180 seconds with the `-t` command line option.

1. Run `cf env` and obtain your database credentials by searching in the `VCAP_SERVICES` environment variable:

```
$ cf env db-app
Getting env variables for app my-db-app in org My-Org / space development as admin...
OK

System-Provided:
{
 "VCAP_SERVICES": {
  "example-db-n/a": [
   {
    "name": "test-777",
    "label": "example-db-n",
    "tags": ["mysql","relational"],
    "plan": "basic",
    "credentials" : {
     "jdbcUrl": "jdbcmysql://aa11:2b@cdbr-05.example.net:3306/ad_01",
     "uri": "mysql://aa11:2b@cdbr-05.example.net:    34/ad_01?reconnect=true",
     "name": "ad_01",
     "hostname": "cdbr-05.example.net",
     "port": "1234",
     "username": "aa11",
     "password": "2b"
    }
   }
  ]
 }
```

2. Connect to the database using your database credentials.

3. Migrate the database using SQL commands.

4. Update the application using `cf push`.

Migrate Occasionally

This method requires you to:

- Create a schema migration command or script.
- Run the migration command when deploying a single instance of the application.
- Re-deploy the application with the original start command and number of instances.

This method is efficient for occasional use because you can re-use the schema migration command or script.

1. Create a schema migration command or SQL script to migrate the database. For example:
   `rake db:migrate`

2. Deploy a single instance of your application with the database migration command as the start command. For example:
   `cf push APP -c 'rake db:migrate' -i 1`

   > **Note**: After this step the database has been migrated but the application itself has not started, because the normal start command is not used.

3. Deploy your application again with the normal start command and desired number of instances. For example:
   `cf push APP -c 'null' -i 4`

   > **Note**: This example assumes that the normal start command for your application is the one provided by the buildpack, which the `-c 'null'` option forces Cloud Foundry to use.

Migrate Frequently

This method uses an idempotent script to partially automate migrations. The script runs on the first application instance only.

This option takes the most effort to implement, but becomes more efficient with frequent migrations.

1. Create a script that:
   - Examines the `instance_index` of the `VCAP_APPLICATION` environment variable. The first deployed instance of an application always has an `instance_index` of "0". For example, this code uses Ruby to extract the `instance_index` from `VCAP_APPLICATION`:
     `instance_index = JSON.parse(ENV["VCAP_APPLICATION"])["instance_index"]`
   - Determines whether or not the `instance_index` is "0".
   - If and only if the `instance_index` is "0", runs a script or uses an existing command to migrate the database. The script or command must be idempotent.

2. Create a manifest that provides:
   - The application name
   - The `command` attribute with a value of the schema migration script chained with a start command.

   Example partial manifest:

```
---
applications:
- name: my-rails-app
  command: bundle exec rake cf:on_first_instance db:migrate && bundle exec rails s -p $PORT -e $RAILS_ENV
```

3. Update the application using `cf push`.

For an example of the migrate frequently method used with Rails, see Running Rake Tasks.

Using an External File System (Volume Services)

This topic describes how Pivotal Cloud Foundry (PCF) app developers can read and write to a mounted file system from their apps. In PCF, a volume service provides a volume so your app can read or write to a reliable, non-ephemeral file system.

Prerequisite

Before you can use a volume service with your app, your Cloud Foundry administrator must add a volume service to your deployment. See the Enabling NFS Volume Services topic for more information.

You can run the Cloud Foundry Command Line Interface (cf CLI) `cf marketplace` command to determine if any volume services are available. See the following example output of the NFS volume service:

```
$ cf marketplace
service   plans      description
nfs       Existing   Service for connecting to NFS volumes
```

If no volume service that fits your requirements exists, contact your Cloud Foundry administrator.

Create and Bind a Service Instance

To use a volume service deployed by your Cloud Foundry administrator, you must first create an instance of the specific volume service that you need. Follow the instructions below to create this service instance.

1. In a terminal window, run `cf create-service SERVICE-NAME PLAN SERVICE-INSTANCE -c SHARE-JSON` to create a service instance. Replace the following with the specified values:
   - `SERVICE` : The name of the volume service that you want to use.
   - `PLAN` : The name of the service plan. Service plans are a way for providers to offer varying levels of resources or features for the same service.
   - `SERVICE-INSTANCE` : A name you provide for your service instance. Use any series of alpha-numeric characters, hyphens, and underscores. You can rename the instance at any time.
   - `SHARE-JSON` (NFS Only): If you create an instance of the NFS volume service, you must supply an extra parameter, `share` , by using the `-c` flag with a JSON string, in-line or in a file. This parameter forwards information to the broker about the NFS server and share required for the service.

   The following example shows creating an instance of the “Existing” NFS service plan, passing an in-line JSON string:

   ```
   $ cf create-service nfs Existing nfs_service_instance -c '{"share": "10.10.10.10/export/myshare"}'
   ```

2. Run `cf bind-service YOUR-APP SERVICE-NAME -c GID-AND-UID-JSON MOUNT-PATH-JSON` to bind your service instance to an app. Replace the following with the specified values:
   - `YOUR-APP` : The name of the PCF app for which you want to use the volume service.
   - `SERVICE-NAME` : The name of the volume service instance you created in the previous step.
   - `GID-AND-UID-JSON` (NFS only): If you bind an instance of the NFS volume service, you must supply two extra parameters, `gid` and `uid` . You can specify these parameters with the `-c` flag and a JSON string, in-line or from a file. This parameter specifies the `gid` and `uid` to use when mounting the share to the app.
   - `MOUNT-PATH` (Optional): To mount the volume to a particular path within your app rather than the default path, you supply the `mount` parameter. Choose a path with a root-level folder that already exists in the container, such as `/home` , `/usr` , or `/var` .

   The following example shows binding `my-app` to the `nfs_service_instance` and specifying the volume to be mounted to `/var/volume1` , passing an in-line JSON string:

   ```
   cf bind-service my-app nfs_service_instance -c '{"uid":"1000","gid":"1000","mount":"/var/volume1"}'
   ```

3. Run `cf restage YOUR-APP` to complete the service binding by restaging your app. Replace `YOUR-APP` with the name of your app.

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

Access the Volume Service from your App

To access the volume service from your app, you must know which file path to use in your code. You can view the file path in the details of the service binding, which are available from the VCAP_SERVICES environment variable. Follow the steps below.

1. Run `cf env YOUR-APP` to view environment variables for your app. Replace `YOUR-APP` with the name of your app.

```
$ cf env my-app
"VCAP_SERVICES": {
"nfs": [
 {
  "credentials": {},
  "label": "nfs",
  "name": "nfs_service_instance",
  "plan": "Existing",
  "provider": null,
  "syslog_drain_url": null,
  "tags": [
  "nfs"
  ],
  "volume_mounts": [
   {
    "container_dir": "/var/vcap/data/153e3c4b-1151-4cf7-b311-948dd77fce64",
    "device_type": "shared",
    "mode": "rw"
   }
  ]
 }
]
}
```

2. Use the properties under `volume_mounts` for any information your app needs. Refer to the following table:

Property	Description
`container_dir`	String containing the path to the mounted volume that you bound to your app.
`device_type`	The NFS volume release. This currently only supports `shared` devices. A `shared` device represents a distributed file system that can mount on all app instances simultaneously.
`mode`	String that informs what type of access your app has to NFS, either read-only, `ro`, or read and write, `rw`.

PCF Security Guide

For Security Professionals and PCF Users

This guide explains how Pivotal Cloud Foundry (PCF) manages network access, roles and permissions, internal communications, container hardening, and other security issues. It is intended to give security professionals a complete view of PCF security, and to help all PCF users, not just the security experts, keep the platform secure.

Current Security Reports

Pivotal publishes security updates regularly in response to privately- and publicly-reported Common Vulnerabilities and Exposures (CVEs).

See the latest CVEs on the Pivotal Application Security Team page.

To learn about Pivotal's vulnerability reporting and responsible disclosure process, read PCF Security Overview and Policy.

Guide Contents

- Securing Traffic into Cloud Foundry - Configuring and maintaining front-end platform security at the load balancer or router.
- Identity Management - Managing permissions and trust for PCF user accounts, and user accounts in the underlying IaaS.
- PCF Component Security - How PCF components and app containers keep internal communications secure, and what paths, ports, and protocols the components use to communicate.
- PAS App and Service Security - Enabling PAS apps to communicate internally with other apps and use service instance credentials securely.
- CredHub - The credential management tool that BOSH uses to store deployment credentials and that PCF runtimes use to create and manage app and service credentials.
- PCF Security and Stemcell Processes - How Pivotal responds to security vulnerabilities, and how it tests and updates the versioned operating systems that its products run on.

Security Concepts

This section provides links to overview and conceptual documentation affecting Pivotal Cloud Foundry (PCF) security requirements.

- Understanding Cloud Foundry Security
  This topic explains the measures Cloud Foundry implements to minimize security risks.
- Understanding Container Security
  This topic explains how PCF isolates and networks containers securely.

Security Processes and Stemcells

This section explains how Pivotal responds to security vulnerabilities, and how it tests and updates its stemcells, the versioned operating systems that its products run on.

- Pivotal Cloud Foundry Security Overview and Policy - Pivotal's responsible disclosure and vulnerability response procedures for the Pivotal Cloud Foundry (PCF) platform.
- PCF Testing, Release, and Security Lifecycle - How Pivotal's practices, tools, and organizational structures work together to create and support stable releases of PCF.
- Understanding Floating Stemcells - How PCF automatically upgrades all compatible products when a new stemcell is available.
- Windows Stemcell Hardening - The settings for Local Group Policy and Local Security Policy that Pivotal incorporates into its Windows stemcells to optimize security.
- Linux Stemcell Hardening - How Pivotal secures Linux stemcells through regular testing and minimizing their surface of vulnerability.

Pivotal Cloud Foundry Security Overview and Policy

Page last updated:

This document outlines our security policy and is addressed to operators deploying Pivotal Cloud Foundry (PCF) using Pivotal Cloud Foundry Operations Manager.

For a comprehensive overview of the security architecture of each PCF component, refer to the Cloud Foundry Security topic.

How Pivotal Monitors for Security Vulnerabilities

Pivotal receives private reports on vulnerabilities from customers and from field personnel via our secure disclosure process. We also monitor public repositories of software security vulnerabilities to identify newly discovered vulnerabilities that might affect one or more of our products.

How to Report a Vulnerability

Pivotal encourages users who become aware of a security vulnerability in our products to contact Pivotal with details of the vulnerability. Please send descriptions of any vulnerabilities found to security@pivotal.io. Please include details on the software and hardware configuration of your system so that we can reproduce the issue.

Note: We encourage use of encrypted email. Our public PGP key is located at http://www.pivotal.io/security.

Notification Policy

PCF has many customer stakeholders who need to know about security updates. When there is a possible security vulnerability identified for a PCF component, we do the following:

1. Assess the impact to PCF.
2. If the vulnerability would affect a PCF component, we schedule an update for the impacted component(s).
3. Update the affected component(s) and perform system tests.
4. Announce the fix publicly via the following channels:
   a. Automated notification to end users who have downloaded or subscribed to a PCF product on Pivotal Network when a new, fixed version is available.
   b. Adding a new post to http://www.pivotal.io/security.

Classes of Vulnerabilities

Attackers can exploit vulnerabilities to compromise user data and processing resources. This can affect data confidentiality, integrity, and availability to different degrees. For vulnerabilities related to Ubuntu provided packages, Pivotal follows Canonical's priority levels. For other vulnerabilities, Pivotal follows Common Vulnerability Scoring System v3.0 standards when assessing severity.

Pivotal reports the severity of vulnerabilities using the following severity classes:

High

High severity vulnerabilities are those that can be exploited by an unauthenticated or authenticated attacker, from the Internet or those that break the guest/host Operating System isolation. The exploitation could result in the complete compromise of confidentiality, integrity, and availability of user data and/or processing resources without user interaction. Exploitation could be leveraged to propagate an Internet worm or execute arbitrary code between Virtual Machines and/or the Host Operating System. This rating also applies to those vulnerabilities that could lead to the complete compromise of availability when the exploitation is by a remote unauthenticated attacker from the Internet or through a breach of virtual machine isolation.

Moderate

Moderate vulnerabilities are those in which the ability to exploit is mitigated to a significant degree by configuration or difficulty of exploitation, but in certain deployment scenarios could still lead to the compromise of confidentiality, integrity, or availability of user data and/or processing resources.

Low

Low vulnerabilities are all other issues that have a security impact. These include vulnerabilities for which exploitation is believed to be extremely difficult, or for which successful exploitation would have minimal impact.

Release Policy

PCF schedules regular monthly releases of software in the PCF Suite to address Low / Medium severity vulnerability exploits. When High severity vulnerability exploits are identified, PCF releases fixes to software in the PCF Suite on-demand, with as fast a turnaround as possible.

Alerts/Actions Archive

http://www.pivotal.io/security

PCF Testing, Release, and Security Lifecycle

Page last updated:

This topic explains how Pivotal's development practices, automated build tools, and organizational structures work together to create and support stable releases of Pivotal Cloud Foundry (PCF).

Summary

- PCF teams building system components receive frequent feedback, which helps to secure code from exposure to vulnerability.
- Every PCF release follows a strict workflow and passes through numerous quality and compliance checks before distribution.
- Teams build tests into the product consistently and run them automatically with any code change.

Release Mechanics

Pivotal releases, patches, and supports multiple versions of PCF simultaneously. This section explains the versioning and support conventions Pivotal follows.

Versioning

Pivotal numbers PCF releases following a semantic versioning style format, *X.Y.Z*, where *X* and *Y* indicate major and minor releases, and *Z* designates patch releases. Major and minor releases change PCF functionality; patch releases are backward-compatible security patches or bug fixes.

Support

As of PCF 1.8, Pivotal supports each major and minor PCF release according to the following criteria:

- Pivotal supports the release for at least 9 months following its first publication date.
- Pivotal supports the last three major or minor releases, even if this extends coverage beyond 9 months.

Support includes maintenance updates and upgrades, bug and security fixes, and technical assistance. The Pivotal Support Policy describes support standards, technical guidance, and publication phases in more detail. The Pivotal Support Services Terms and Conditions defines Pivotal support in legal terms.

Patch Releases

Patch releases are more frequent and less predictable than major/minor releases. The v1.6.x line provides a good example of their frequency. PCF 1.6.1 was released on October 26, 2015. Through August 2016, 36 additional patches of Elastic Runtime 1.6.x and 18 patches of Ops Manager 1.6.x provided security and bug fixes to customers.

Pivotal identifies security issues using standard nomenclature from Common Vulnerabilities and Exposures (CVE), Ubuntu Security Notices (USN), and other third party sources. Read about security fixes in core Cloud Foundry code or packaged dependencies in the release notes for Ops Manager and Pivotal Application Service (PAS).

Pivotal.io/security maintains a running list of security fixes in PCF and PCF dependencies. Consult that page to see the most recent findings from Pivotal's security team.

Upgrading

All PCF releases pass through extensive test suites that include automated unit, integration, and acceptance tests on multiple IaaSes. Regardless of this extensive testing, Pivotal recommends that you test major and minor releases in a non-production environment before implementing them across your deployment. Upgrade your production environment as soon as possible after you validate the new release on your test environment.

Release Testing, Integration, and Validation

This section describes Pivotal's software development processes and explains compliance and regulatory standards to which Pivotal software adheres.

Test-Driven Development

Pivotal's development process relies on a strict workflow with continuous automated testing. Pivotal R&D does not separate engineering and testing teams. Rather, every Pivot on each engineering team is responsible for ensuring the quality of their code. They write tests for all of the software components that they develop, often before writing the software itself.

With every software change, automated build pipelines trigger these tests for the new software component and for everything it touches. If a new code check-in does not pass its tests or causes a failure elsewhere, it pauses the build pipeline for the entire team, or sometimes all of Pivotal R&D. The transparency of this process encourages developers to work together to address code issues quickly.

Pivotal applies the following automated testing approaches, scenarios, and frameworks to PCF components and to the release as a whole:

- **Unit tests**: Development teams write unit tests to express and validate desired functional behavior of product components. Typical frameworks used are RSpec and Ginkgo. These tests run continuously throughout the development cycle.
- **OSS integration tests**: The Release Integration team exercises a full deployment of open-source Cloud Foundry to validate all end-user features. They maintain the Cloud Foundry Acceptance Test (CATs) suite alongside the OSS cf-release. Cloud Foundry component teams also contribute acceptance test suites at the OSS Integration Test level. These tests exercise and validate their components' functional, performance, and integration health.
- **PCF integration tests**: The PCF Release Engineering (RelEng) team validates the quality and cross-product integration health of the commercial PCF release. RelEng runs OSS Acceptance Tests against all supported releases. These tests run on full PCF instances configured to represent diverse real-world customer scenarios on various IaaSes and using both internal and external load balancer, database, blobstore, and user store solutions.

Additional Pre-Release Gates: Internal, PWS, and Compliance

In addition to its automated unit and integration testing, Pivotal deploys all upgrades slated for upcoming PCF releases on at-scale test environments. Prior to each Major or Minor commercial release, Pivotal runs the entire Pivotal Cloud Foundry Suite of services on several internally-managed large integration environments that run customer-like data and workloads.

Pivotal also pushes upcoming PCF feature upgrades and patches to its Pivotal Web Services platform, where customers continually deploy and host hundreds of mission-critical applications at scale, 24/7. The PWS environment gives Pivotal a continuous source of real-world usage and performance metrics that inform product development teams.

All PCF product teams participate in go-to-market steps for each release, as is often required for shipping a legally compliant product. Examples include Open Source License File attribution and an Export Compliance classification.

Patch Releases: Security and Bug Fixes

Pivotal uses established processes to track, disclose, and remediate vulnerabilities in PCF and related dependent components. This section explains how Pivotal identifies vulnerabilities and implements fixes for them.

Identifying Security Vulnerabilities

Pivotal has an established process to track and patch vulnerabilities in software dependencies and PCF software. Additionally, pivotal.io/security describes a responsible disclosure process for reporting vulnerabilities identified in Pivotal software by 3rd parties.

Pivotal uses multiple methods to identify security vulnerabilities in Pivotal software and dependencies internally, including:

- Security notifications from Canonical for their Ubuntu operating system, provided through Pivotal's commercial relationship with Canonical
- Software component scans several times per day, using 3rd party security software which updates continuously from external security vulnerability sources
- Dependency analysis software that identifies and catalogs software dependencies
- Security vulnerability notifications from known software dependencies

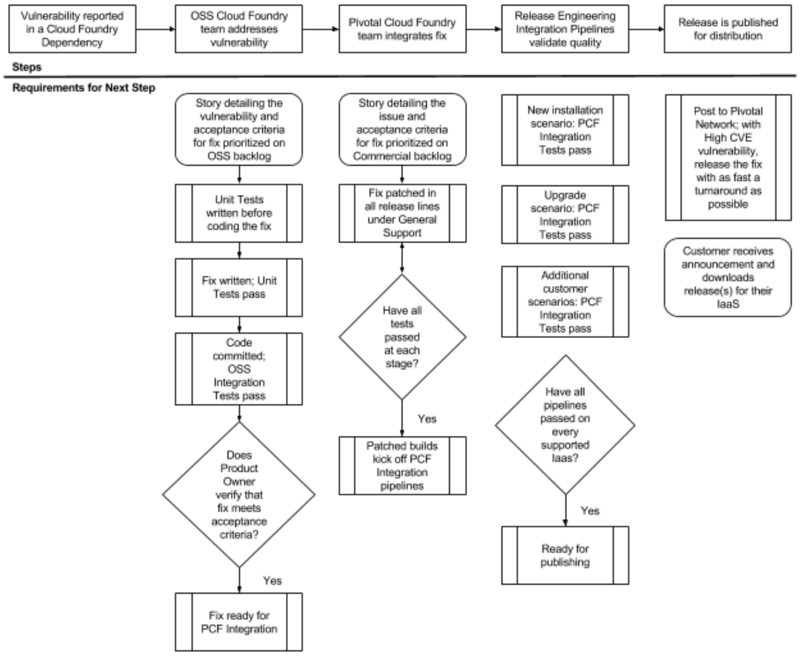
Pivotal also monitors externally-reported vulnerabilities from many sources, including:

- 3rd party security analysis requested by Pivotal
- Cloud Foundry Foundation security notifications from member companies
- Customer, prospect and other 3rd party security reports

When Pivotal discovers a potential security vulnerability in PCF, the security team opens an issue to assess it. If it confirms the vulnerability exists, Pivotal identifies and updates affected components with plans to backport the fix to stable releases. Fixes are implemented on a target timeline based on the severity level of the vulnerability.

Fix, Test, and Release Lifecycle

This flowchart details the steps that Pivotal performs on a typical high-priority CVE, to publish a patch release fix on https://network.pivotal.io:



Identity Management

This section provides links to different aspects of identity management, including user creation and permissions management, authentication, and event logging for Pivotal Cloud Foundry (PCF).

- Creating and Managing Users with the UAA CLI (UAAC)
- Security Event Logging for Cloud Controller and UAA

PCF App and Service Security

This section describes how PCF and PCF users manage security for apps and service instances.

- Configuring SSH Access for PCF - How to configure PCF to allow SSH access to app instances, for debugging.
- Understanding Container-to-Container Networking - How to create app-specific policies that enable internal app-to-app communication.
- Restricting App Access to Internal PCF Components - How to create Application Security Groups (ASGs), rules that allow internal outgoing communications from all apps in PCF, or the apps running in the same space.
- Understanding Application Security Groups - How ASGs work and how to manage them in all versions of Cloud Foundry, including PCF.
- Configuring Application Security Groups for Email Notifications - How to define an ASG to enable app-generated notifications.
- Trusted System Certificates - Where applications can find trusted system certificates.
- Delivering Service Credentials to an Application - How binding apps to a service instance generates credentials that enable the apps to use the service.
- Managing Service Keys - How to create and manage service keys that enable apps to use service instances.

Pivotal

Managing Non-Configurable TLS/SSL Certificates

This topic describes how to manage Certificate Authority (CA) certificates in your deployment, including how to determine when your CA certificates expire, how to set a custom CA certificate, and how to regenerate and rotate your CA certificates.

For information about rotating IPsec certificates, see Rotating IPsec Certificates.

Planning for Certificate Expiration

Non-configurable CA certificates expire 2 years after creation. Root certificates expire after 4 years. These asychronous lifecycles make it possible for you to rotate non-configurable certificates without creating and applying a new root certificate.

Depending on the needs of your deployment, there are two approaches to certificate rotation:

- You can only rotate non-configurable certificates, or
- You can create and apply new root and non-configurable certificates.

Rotating Non-Configurable Certificates

If you need to rotate non-configurable certificates, follow the procedure below. If you need to rotate all your certificates, including creating and applying a new root certificate, follow the procedure in Regenerate and Rotate CA Certificates.

Rotating Expiring Certificates

1. Use `curl` to make an API call to regenerate all non-configurable certificates and apply the new CA to your existing Ops Manager Director:

   ```
   $ curl "https://OPS-MAN-FQDN/api/v0/certificate_authorities/active/regenerate" \
     -X POST \
     -H "Authorization: Bearer YOUR-UAA-ACCESS-TOKEN" \
     -H "Content-Type: application/json" \
     -d '{}'
   ```

   The API returns a successful response:

   ```
   HTTP/1.1 200 OK
   ```

2. Navigate to Ops Manager and click **Apply Changes**. When the deploy finishes, continue to the next section.

Generate and Retrieve Certificates

To manage and retrieve information about certificates in your deployment, use the API calls in this section.

Perform the steps in the Using Ops Manager API topic to target and authenticate with the Ops Manager User Account and Authentication (UAA) server. Record your Ops Manager access token, and use it for `YOUR-UAA-ACCESS-TOKEN`.

Retrieve the Root CA Certificate

To return the Ops Manager root CA certificate as a file, use `curl` to make the following API call:

```
$ curl "https://OPS-MAN-FQDN/api/v0/download_root_ca_cert \
      -X GET \
      -H "Authorization: Bearer YOUR-UAA-ACCESS-TOKEN"
```

To return the Ops Manager root CA certificate as JSON, use `curl` to make the following API call:

```
$ curl "https://OPS-MAN-FQDN/api/v0/security/root_ca_certificate \
```

```
     -X GET \
     -H "Authorization: Bearer YOUR-UAA-ACCESS-TOKEN"
```

Retrieve and Generate RSA Certificates

To generate and return a new RSA certificate signed by the root CA, use `curl` to make the following API call:

```
$ curl "https://OPS-MAN-FQDN/api/v0/certificates/generate \
     -X POST \
     -H "Authorization: Bearer YOUR-UAA-ACCESS-TOKEN"
```

To return metadata from all deployed RSA certificates signed by the root CA, use `curl` to make the following API call:

```
$ curl "https://OPS-MAN-FQDN/api/v0/deployed/certificates \
     -X GET \
     -H "Authorization: Bearer YOUR-UAA-ACCESS-TOKEN"
```

Check Certificate Expiration Dates

The non-configurable certificates in your deployment expire every two years. Use the following procedure to retrieve information about the expiration dates for RSA and CA certificates in your deployment.

1. Perform the steps in the Using Ops Manager API topic to target and authenticate with the Ops Manager User Account and Authentication (UAA) server. Record your Ops Manager access token, and use it for `YOUR-UAA-ACCESS-TOKEN`.

2. To check the system for certificates that expire within a given time interval, use `curl` to make an API call.

   Use the `https://OPS-MAN-FQDN/api/v0/deployed/certificates?expires_within=TIME` endpoint, replacing `TIME` with an integer and a letter code. Valid letter codes are `d` for days, `w` for weeks, `m` for months, and `y` for years.

   For example, run the following command to search for certificates expiring within six months:

   ```
   $ curl "https://OPS-MAN-FQDN/api/v0/deployed/certificates?expires_within=6m" \
        -H "Authorization: Bearer YOUR-UAA-ACCESS-TOKEN"
   ```

   Replace `YOUR-UAA-ACCESS-TOKEN` with the `access_token` value you recorded in the previous step.

Regenerate and Rotate CA Certificates

Warning: You must complete the procedures in this topic in the exact order specified below. Otherwise, you risk doing damage to your deployment.

Depending on the requirements of your deployment, you may need to rotate your non-configurable CA certificates. Certificates can expire or fall out of currency, or your organization's security compliance policies may require you to rotate certificates periodically.

You can rotate the certificates in your Pivotal Cloud Foundry (PCF) deployment using `curl`. PCF provides different API calls with which to manage certificates and CAs. New certificates generated through this process use SHA-256 encryption.

These API calls allow you to create new CAs, apply them, and delete old CAs. The process of activating a new CA and rotating it in gives new certificates to the Ops Manager Director. The BOSH Director then passes the certificates to other components in your PCF deployment.

Note: These procedures require you to return to Ops Manager and click **Apply Changes** periodically. Clicking **Apply Changes** redeploys the Ops Manager Director and its tiles. If you apply your changes during each procedure, a successful redeploy verifies that the certificate rotation process is proceeding correctly.

Step 1: Add a New CA

Use the following procedure to add a new CA. You can use a Pivotal-generated CA or provide your own CA using the API calls in this section.

1. Perform the steps in the Using Ops Manager API topic to target and authenticate with the Ops Manager User Account and Authentication (UAA) server. Record your Ops Manager access token, and use it for `YOUR-UAA-ACCESS-TOKEN` in the following procedures.

   > **Note**: When you record your Ops Manager access token, make sure you remove any new line characters such as `/n`. Otherwise the API call in the following step will not succeed.

2. To use a Pivotal-generated CA, use `curl` to make the following API call:

```
$ curl "https://OPS-MAN-FQDN/api/v0/certificate_authorities/generate" \
  -X POST \
  -H "Authorization: Bearer YOUR-UAA-ACCESS-TOKEN" \
  -H "Content-Type: application/json" \
  -d '{}'
```

   If the command succeeds, the API returns a response that includes the new certificate.


```
HTTP/1.1 200 OK
{
"guid": "f7bc18f34f2a7a9403c3",
"issuer": "Pivotal",
"created_on": "2017-01-19",
"expires_on": "2021-01-19",
"active": false,
"cert_pem": "-----BEGIN EXAMPLE CERTIFICATE-----
 MIIC+zCCAeOgAwIBAgIBADANBgkqhkiG9w0BAQsFADAfMQswCQYDVQQGEwJVUzEQ
 MA4GA1UECgwHUGl2b3RhbDAeFw0xNzAxMTgyMTQyMjVaFw0yMTAxMTkyMTQyMjVa
 MB8xCzAJBgNVBAYTAlVTMRAwDgYDVQQKDAdQaXZvdGFsMIIBIjANBgkqhkiG9w0B
 AQEFAAOCAQ8AMIIBCgKCAQEAyV4OhPIlZTEym9OcdcNVip9Ev0ijPPLo9WPLUMzT
 IrpDx3nG/TgD+DP09mwVXfqwBlJmoj9DqRED1x/6bc0Ki/BAFo/P4MmOKm3QnDCt
 o+4RUvLkQqgA++2HYrNTKWJ5fsXmERs8lK9AXXT7RKXhktyWWU3oNGf7zo0e3YKp
 l07DdIW7h1NwIbNcGT1AurIDsxyOZy1HVzLDPtUR2MxhJmSCLsOw3qUDQjatjXKw
 82RjcrswjG3nv2hvD4/aTOiHuKM3+AGbnmS2MdIOvFOh/7Y79tUp89csK0gs6uOd
 myfdxzDihe4DcKw5CzUTfHKNXgHyeoVOBPcVQTp4lJp1iQIDAQABo0IwQDAdBgNV
 HQ4EFgQUyH4y7VEuImLStXM0CKR8uVqxX/gwDwYDVR0TAQH/BAUwAwEB/zAOBgNV
 HQ8BAf8EBAMCAQYwDQYJKoZIhvcNAQELBQADggEBALmHOPxdyBGnuR0HgR9V4TwJ
 tnKFdFQJGLKVT7am5z6G2Oq5cwACFHWAFfrPG4W9Jm577QtewiY/Rad/PbkY0YSY
 rehLThKdkrfNjxjxI0H2sr7qLBFjJ0wBZHhVmDsO6A9PkfAPu4eJvqRMuL/xGmSQ
 tVkzgYmnCynMNz7FgHyFbd9D9X5YW8fWGSeVBPPikcONdRvjw9aEeAtbGEh8eZCP
 aBQOgsx7b33RuR+CTNqThXY9k8d7/7ba4KVdd4gP8ynFgwvnDQOjcJZ6Go5QY5HA
 R+OgIzs3PFW8pAYcvWrXKR0rE8fL5o9qgTyjmO+5yyyvWIYrKPqqIUIvMCdNr84=
 -----END EXAMPLE CERTIFICATE-----
 "
```


   To provide your own CA, use `curl` to make the following API call:

```
$ curl "https://OPS-MAN-FQDN/api/v0/certificate_authorities" \
    -X POST \
    -H "Authorization: Bearer YOUR-UAA-ACCESS-TOKEN" \
    -H "Content-Type: application/json" \
    -d '{"cert_pem": "-----BEGIN CERTIFICATE-----\EXAMPLE-CERTIFICATE", "private_key_pem": "-----BEGIN EXAMPLE RSA PRIVATE KEY-----\EXAMP
```

   Replace `EXAMPLE-CERTIFICATE` with your custom CA certificate and `EXAMPLE-KEY` with your custom RSA key.

3. Confirm that your new CA has been added by listing all of the root CAs for Ops Manager:

```
$ curl "https://OPS-MAN-FQDN/api/v0/certificate_authorities" \
    -X GET \
    -H "Authorization: Bearer YOUR-UAA-ACCESS-TOKEN"
```

   The API call returns the following:

```
HTTP/1.1 200 OK
{
 "certificate_authorities": [
   {
    "guid": "f7bc18f34f2a7a9403c3",
    "issuer": "Pivotal",
    "created_on": "2017-01-09",
    "expires_on": "2021-01-09",
    "active": true,
    "cert_pem": "-----BEGIN CERTIFICATE-----\nMIIC+zCCAeOgAwIBAgI....etc"
   }
   {
    "guid": "a8ee01e33e3e3e3303e3",
    "issuer": "Pivotal",
    "created_on": "2017-04-09",
    "expires_on": "2021-04-09",
    "active": false,
    "cert_pem": "-----BEGIN CERTIFICATE-----\zBBBC+eAAAe1gAwAAAeZ....etc"
   }
 ]
}
```


Identify your newly added CA, which has `active` set to `false`. Record its GUID.

4. Navigate to `https://OPS-MAN-FQDN` in a browser and log in to Ops Manager.

5. Click **Apply Changes**. When the deploy finishes, continue to the next section.

Step 2: Activate the New CA

1. Use `curl` to make an API call to activate the new CA, replacing `CERT-GUID` with the GUID of your CA that you retrieved in the previous section:

```
$ curl "https://OPS-MAN-FQDN/api/v0/certificate_authorities/CERT-GUID/activate" \
  -X POST \
  -H "Authorization: Bearer YOUR-UAA-ACCESS-TOKEN" \
  -H "Content-Type: application/json" \
  -d '{}'
```

The API returns a successful response:

```
HTTP/1.1 200 OK
```

2. List your root CAs to confirm that the new CA is active:

```
$ curl "https://OPS-MAN-FQDN/api/v0/certificate_authorities" \
    -X GET \
    -H "Authorization: Bearer YOUR-UAA-ACCESS-TOKEN"
```

Examine the response to ensure that your new CA has `active` set to `true`.

Step 3: Regenerate Non-Configurable Certificates to Apply the New CA

1. Use `curl` to make an API call to regenerate all non-configurable certificates and apply the new CA to your existing Ops Manager Director:

```
$ curl "https://OPS-MAN-FQDN/api/v0/certificate_authorities/active/regenerate" \
    -X POST \
    -H "Authorization: Bearer YOUR-UAA-ACCESS-TOKEN" \
    -H "Content-Type: application/json" \
    -d '{}'
```

The API returns a successful response:

```
HTTP/1.1 200 OK
```

2. Navigate to Ops Manager and click **Apply Changes**. When the deploy finishes, continue to the next section.

Step 4: Delete the Old CA

1. List your root CAs to retrieve the GUID of your old, inactive CA:

```
$ curl "https://OPS-MAN-FQDN/api/v0/certificate_authorities" \
    -X GET \
    -H "Authorization: Bearer YOUR-UAA-ACCESS-TOKEN"
```

2. Use `curl` to make an API call to delete your old CA, replacing `OLD-CERT-GUID` with the GUID of your old, inactive CA:

```
$ curl "https://OPS-MAN-FQDN/api/v0/certificate_authorities/:OLD-CERT-GUID" \
    -X DELETE \
    -H "Authorization: Bearer YOUR-UAA-ACCESS-TOKEN"
```

   The API returns a successful response.

```
HTTP/1.1 200 OK
```

3. Navigate to Ops Manager and click **Apply Changes**.

Pivotal

Linux Stemcell Hardening

Note: This document applies to stemcell v3263.

Customers and prospects often ask for details on stemcell hardening, i.e., the process by which we secure Pivotal Cloud Foundry by reducing its vulnerability surface from outside access. This document provides responses to some commonly-asked questions regarding the security configuration enhancements and hardening tests that Pivotal applies to the Cloud Foundry ("CF") stemcell. This information will be helpful to customer accreditation teams who are responsible for running configuration scans of a Cloud Foundry deployment, and also to auditors who need a documentation artifact to feed into the customers' existing security assessment processes.

1. **WHAT IS A STEMCELL?** A stemcell is a versioned Operating System ("OS") image wrapped with IaaS specific packaging. A typical stemcell contains a bare minimum OS skeleton with a few common utilities pre-installed, a BOSH Agent, and a few configuration files to securely configure the OS by default. For example: with vSphere, the official stemcell for Ubuntu Trusty is an approximately 500MB VMDK file. With AWS, official stemcells are published as AMIs that can be used in an AWS account. Stemcells do not contain any specific information about any software that will be installed once that stemcell becomes a specialized machine in the cluster; nor do they contain any sensitive information which would make them unable to be shared with other BOSH users. This clear separation between base OS and later-installed software is what makes stemcells a powerful concept. In addition to being generic, stemcells for one OS (e.g. all Ubuntu Trusty stemcells) are exactly the same for all infrastructures. This property of stemcells allows BOSH users to quickly and reliably switch between different infrastructures without worrying about the differences between OS images. The CF BOSH team is responsible for producing and maintaining an official set of stemcells. Cloud Foundry currently supports Ubuntu Trusty on vSphere, AWS, OpenStack, Google, and Azure infrastructures.

2. **WHAT IS STEMCELL HARDENING?** Stemcell hardening is the process of securing a stemcell by reducing its surface of vulnerability, which is larger when a system performs more functions; in principle a single-function system is more secure than a multipurpose one. There are various methods of hardening Linux systems. Common techniques include reducing available methods of attack by implementing more restrictive and/or conservative configurations of the OS kernel and system services, changing default passwords, the removal of unnecessary software, unnecessary usernames and logins, and the disabling or removal of unnecessary services.

3. **WHAT IS OUR GENERAL APPROACH TO STEMCELL HARDENING?** The CF stemcell is essentially a distinct Linux distribution. As such, industry-standard benchmarks are not entirely appropriate when assessing the security posture of the stemcell, but Pivotal has considered and incorporated hardening guidance from various sources both commercial and government. Some parts of the existing recommended industry-standard hardening configurations will certainly apply, but some other parts do not apply. In addition, because each stemcell is a unique Linux distribution, existing industry-standard benchmarks are silent on some important aspects of hardening the stemcell configurations. The following paragraphs describe the different categories of stemcell hardening configurations, and provide a count of the number of tests currently in each category. **Note**: The most current description of what has been delivered is always available in the BOSH public Pivotal Trackers.

    a. **Baseline Passing:** common hardening tests that pass without any changes to the stemcell or to test procedures. *(130 tests)*
    b. **Test Amended:** Stemcells are optimized for cloud deployment and some configuration settings are not stored in traditionally-expected locations. The industry standard test was changed to conform with stemcell design to accurately check the recommended setting. This new test reflects the changes to the industry standard test but the stemcell adheres to commonly accepted guidance. *(36 tests)*
    c. **Additional Hardening:**Configuration hardening improvements that have been made to the stemcell. As with most software, a stemcell's security improves over time and every stemcell release is tested to ensure that it is suitable for use with its associated CF release. Later releases of a stemcell may include additional security features that were not present in earlier releases. *(86 tests)*
    d. **New CF-specific Tests:** New tests that have been added to check CF stemcell-specific configurations. These tests are not yet part of any industry standard Ubuntu benchmark. This category of tests is still under development and additional tests will be added over time. *(20 tests)*

4. **WHAT ARE THE MAJOR FOCUS AREAS FOR OUR STEMCELL HARDENING APPROACH?**

    a. **Maintenance, Updates, and Patching**

        i. Regular patches and feature enhancements are delivered via routine BOSH deployments of updated stemcells (obviates apt-get upgrade).

    b. **File System Hardening**

        i. The /tmp directory is configured to be on a separate partition.
        ii. Users cannot create character or block special devices in the /tmp filesystem.
        iii. Users cannot create set userid files in the /tmp filesystem.
        iv. Users cannot run executable binaries from the /tmp filesystem.
        v. The temporary storage directories such as /tmp and /var/tmp are mounted on a dedicated partition, and configured with appropriately limiting options such as nodev, nosuid, and noexec.
        vi. Each of the following directories is in a separate partition, with mount options managed via BOSH agent:
            - /var
            - /var/log
            - /var/log/audit
            - /home
            - /tmp

vii. File system mount options for users' home directories are limited via appropriate mount options including nodev.
viii. Removable media may not be mounted as character or block special device.
ix. Executable programs may not run from removable media.
x. setuid and setgid are not allowed on removable media.
xi. Users cannot create special devices in shared memory partitions.
xii. Users cannot put privileged programs onto shared memory partitions.
xiii. Users cannot execute programs from shared memory partitions.
xiv. Users cannot delete or rename files in world-writable directories such as /tmp that are owned by other users.
xv. Supplementary and exotic Linux file systems that are unused in CF have been disabled.
xvi. Additional supplementary and exotic Linux file systems that are unused in CF have been disabled.
xvii. Automount of USB drives or disks is not permitted.

c. **Boot Security**

i. The owner and group for the bootloader config (/boot/grub/grub.cfg) is set to root. Only root has read and write access to this file.
ii. Boot loader has been configured so that a password is required to reboot the system.
iii. Unauthorized users cannot reboot the system into single user mode.

d. **Process Security**

i. Users cannot override the soft limit for core dumps.
ii. Randomized virtual memory region placement is enabled.
iii. Prelinking of shared libraries is disabled.

e. **Minimization of Attack Surface**

i. The Network Information Service ("NIS") is not used in CF and is not installed.
ii. The Berkeley rsh-server package is not used in CF and is disabled.
iii. Classic rsh-related tools are not used in CF and are not installed.
iv. The following servers are not used on CF stemcells and are disabled:
   - talk server
   - telnet server
   - tftp-server
   - Avahi
   - print
   - DHCP
   - DNS
   - FTP
   - IMAP
   - POP
   - HTTP
   - SNMP

v. The talk client is not used in CF and is not installed.
vi. The eXtended InterNET Daemon (xinetd) is not used in CF and is disabled.
vii. The following network services are not used in CF and are disabled:
   - chargen
   - daytime
   - echo
   - discard
   - time

viii. The X Window system is not used in CF and is not installed.
ix. NTP time setting is synchronized on the stemcell via the ntpdate utility.
x. The Samba daemon is not used in CF and is disabled.
xi. The Mail Transfer Agents (MTA) process only local mail.
xii. The rsync service is not used in CF and is disabled.
xiii. The biosdevname tool is disabled.

f. **Network Security**

i. IPv4 networking is configured such that IP forwarding is disabled.
ii. The IPv4 networking has been configured such that the host cannot send ICMP redirects.
iii. IPv4 networking has been configured such that the system does not accept source routed packets.
iv. IPv4 networking is configured such that ICMP redirects are not accepted.
v. ICMP echo and timestamp requests with broadcast or multicast destinations will be ignored.
vi. The stemcell will ignore malformed ICMP error responses.
vii. IPv4 networking is configured for source route validation.
viii. TCP SYN cookies are enabled.

ix. Stemcells are set to refuse IPv6 router advertisements.
x. The /etc/hosts.allow file exists and is empty.
xi. The /etc/hosts.allow and /etc/hosts.deny files are protected from unauthorized write access.
xii. The /etc/hosts.deny file exists and is empty.
xiii. The following protocols are not used in CF and are disabled:
    - SCTP
    - DCCP
    - TIPC
    - LDAP
    - RDS
xiv. Wireless interfaces are disabled.
xv. IPv6 is not used in CF deployments and the IPv6 protocol is disabled.

g. **Auditing**

i. Audit log file size is configured for a manageable maximum size of 6 MB.
ii. The system auditd logs have been configured such that the system is resilient in the event of a denial of service attack on the auditd daemon.
iii. Auditd daemon is configured such that all auditd logs are kept after rotation.
iv. The auditd service is enabled.
v. Auditing of successful and failed login/logout events is enabled.
vi. The Linux auditing subsystem has been configured in accordance with best practice industry guidance to capture all security-relevant events. The /etc/audit/audit.rules configuration now contains more than 50 monitoring rules.
vii. Audit records are created for loading and unloading of kernel modules and for system calls.
viii. File Integrity Monitoring can be done on the stemcell (via a BOSH Add-on).

h. **Authentication and Authorization**

i. The cron daemon is enabled.
ii. Access to the /etc/crontab file is limited to root.
iii. Access to the cron utility configuration via the hourly, daily, weekly, and monthly directories is limited.
iv. User authorization to schedule cron jobs is limited.
v. Only the vcap user is whitelisted to use the cron and at utilities.
vi. Password requirements follow industry best practice guidance and enforce a minimum length of 14 characters, with at least one each of: digit, uppercase, lowercase and special characters.
vii. Password reuse: users cannot reuse their twenty most recent passwords.
viii. SSH protocol version is configured for SSH-2.
ix. Logging level for SSH event is INFO.
x. Minimum permissions are set on /etc/ssh/sshd_config.
xi. SSH X11 forwarding is disabled.
xii. The MaxAuthTries parameter for SSH is set to 3 attempts per connection.
xiii. SSH is configured to require passwords and ignore host-based authentication.
xiv. Root logins are not allowed over SSH.
xv. Users cannot set environment variables through the SSH daemon.
xvi. SSH has been configured to use strong ciphers:
    - aes128-ctr
    - aes192-ctr
    - aes256-ctr
xvii. Idle SSH sessions are terminated after 15 minutes, and no client “keep alive” messages are sent.
xviii. Idle SSH sessions are terminated after 15 minutes. No client “keep alive” messages are sent.
xix. The SSH login banner may be configured to display site-specific text before user authentication is permitted (via BOSH Add-on).
xx. Root login is only permitted via console, not via tty devices.
xxi. Only the vcap user is authorized in the sudo group.
xxii. Only users in the root group (a.k.a. wheel) are authorized to run the su command.

i. **Compliance**

i. Contents of /etc/issue and /etc/issue.net have been configured to the phrase: “‘Unauthorized use is strictly prohibited. All access and activity is subject to logging and monitoring.” This may be amended if and as necessary via a BOSH Add-on.
ii. The Message of the Day file /etc/motd is not used, but may be populated via a BOSH Add-on if needed.
iii. Identification of the OS and/or version information about the OS does not appear in any login banners.

j. **File System Permissions**

i. The /etc/passwd, /etc/shadow, and /etc/group files are protected from unauthorized write access.
ii. Use and/or presence of any world-writable files has been audited, and minimized to the extent possible for CF.
iii. By default, all stemcell files are owned by a known user and group, and may not belong to a non-existent user or group.

iv. Use of SUID and GUID is restricted, and only the /usr/bin/sudo and /bin/su programs are authorized as SUID and/or GUID programs.

k. **User Account Management**

i. Users cannot change their password more than once a day.
ii. Users are notified 7 days before their passwords expire.
iii. Interactive logins are disabled for system accounts.
iv. The GID for the root account is 0.
v. User accounts may not have empty passwords.
vi. NIS is not used in CF, and integration of OS security configuration with legacy NIS permissioning is not enabled (e.g., for /etc/passwd, shadow, and group).
vii. By default, the only UID 0 account present is root.
viii. By default, the root PATH does not include any risky directory such as the current working (.) or any writable directory.
ix. Minimum privileges are applied to all users' hidden configuration ("dot") files.
x. The .netrc and .rhosts and .forward files are not used in CF and are not present in any user home directory.
xi. Any group present in the /etc/passwd file must also exist in the /etc/group file.
xii. Users defined in /etc/password must have a valid home directory.
xiii. Users must own their home directories.
xiv. All references to user and group names, as well as UID and/or GID identifiers, are self consistent, with no duplicates or orphans allowed.
xv. By default, the shadow group is not used in CF and must be empty.

Network Security

This section introduces some of the networking and routing security options for your Pivotal Cloud Foundry (PCF) deployment.

Securing Traffic and Controlling Routes

You can enable and configure a number of customization options to secure traffic in and out of your PCF deployment.

- TLS Connections in PCF Deployments
- Securing Traffic into Cloud Foundry
- Providing a Certificate for Your SSL/TLS Termination Point
- Enabling TCP Routing

Using the IPsec Add-On

The IPsec add-on for PCF provides additional security to the network layer for each BOSH-deployed virtual machine (VM).

The PCF IPsec add-on secures network traffic within a Cloud Foundry deployment and provides internal system protection if a malicious actor breaches your firewall.

- Securing Data in Transit with the IPsec Add-on
- Rotating IPsec Credentials
- Installing the Pivotal Cloud Foundry IPsec Add-On

Network Communication Paths in PCF

- Cloud Controller Network Communications
- Consul Network Communications
- Container-to-Container Network Communications
- CredHub Network Communications
- Diego Network Communications
- Loggregator Network Communications
- MySQL Network Communications
- NATS Network Communications
- Routing Network Communications
- UAA Network Communications

PCF Component and Container Security

Pivotal Cloud Foundry (PCF) uses Transport Layer Security (TLS) protocols to secure connections between internal components, app containers, and customer hardware.

Within a PCF deployment, TLS secures connections between components like the Ops Manager Director and service tiles. PCF components also use TLS connections to secure communications with external hardware, such as customer load balancers.

In Pivotal Application Service (PAS), app instance containers have identity credentials that enable TLS communication by app instances.

App Instance Container Identity Credentials

Each app instance container in PCF has a PEM-encoded X.509 certificate and PKCS #1 RSA private key. App developers can use these credentials to enable secure TLS communications from their apps.

Instance container certificates are issued from the platform root Certificate Authority (CA) and have the following property values:

- **Common Name** is the app instance GUID
- **Organizational Unit** is the app GUID
- **Subject Alternative Name** (SAN) contains the container IP address for the app instance

Each container stores its credentials in the following variables:

Credential / Keypair Element	Environment Variable	Command to Retrieve Value
Certificate	`CF_INSTANCE_CERT`	`cf ssh APP-NAME -c 'cat $CF_INSTANCE_CERT'`
Private Key	`CF_INSTANCE_KEY`	`cf ssh APP-NAME -c 'cat $CF_INSTANCE_KEY'`

Each container VM also stores the platform root CA certificate in `/etc/cf-system-certificates`.

To run TLS connections or otherwise use instance container credentials from their apps, developers must first add the certificates to their development stack configuration.

For more information about instance ID creds, see the Instance Identity document in the diego-release repository.

TLS Cipher Suites

By default, PCF uses a limited set of cipher suites to secure its internal communications. However, some components used in PCF, like the Gorouter and HAProxy, may support additional TLS cipher suites to accommodate older clients outside of PCF.

> The AWS Classic load balancer does not support the recommended TLS cipher suites. See Securing Traffic into Cloud Foundry for details and mitigations.

For components that allow you to configure TLS cipher suites, only specify the TLS cipher suites that you need.

TLS Cipher Suite Recommendations

The default and recommended version of TLS to use is TLS v1.2.

The recommended TLS cipher suites to use within PCF are the following:

- TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

Gorouter Configuration

As part of your PAS networking configuration, you must specify the TLS cipher suites that the Gorouter uses to secure its communications. Only specify the cipher suites that you need.

The recommended TLS cipher suites for the Gorouter are:

- ECDHE-RSA-AES128-GCM-SHA256
- ECDHE-RSA-AES256-GCM-SHA384

You can specify other cipher suites and a different minimum version of TLS support if your deployment requires it. For a list of other cipher suites and other versions of TLS that are optionally supported by the Gorouter, see Securing Traffic into Cloud Foundry.

For instructions on how to configure the TLS cipher suites for the Gorouter, see the PAS installation documentation for the IaaS of your deployment. For example, if you are deploying PAS on GCP, see Step 6: Configure Networking.

HAProxy Configuration

As part of your PAS networking configuration, you must specify the TLS cipher suites that HAProxy uses to secure its communications. Only specify the cipher suites that you need.

The recommended TLS cipher suites for HAProxy are:

- DHE-RSA-AES128-GCM-SHA256
- DHE-RSA-AES256-GCM-SHA384
- ECDHE-RSA-AES128-GCM-SHA256
- ECDHE-RSA-AES256-GCM-SHA384

You can specify other cipher suites and a different minimum version of TLS support if your deployment requires it. For a list of other cipher suites and other versions of TLS that are optionally supported by HAProxy, see ciphers - Cipher Suite Names in the OpenSSL documentation.

If you use the default and recommended Gorouter TLS cipher suites in PAS, then ensure you have included these Gorouter TLS cipher suites in your HAProxy TLS cipher suite configuration.

If you change the default Gorouter TLS cipher suites in PAS, and you change the TLS cipher suites for HAProxy, ensure that you have at least one overlapping TLS cipher suite within the two sets.

For instructions on how to configure the TLS cipher suites for HAProxy, see the PAS installation documentation for the IaaS of your deployment. For example, if you are deploying PAS on GCP, see Step 6: Configure Networking.

Component Communications

The following topics describe how network communications work for main subsystem of PCF. Each topic lists the paths, ports, and protocols that the components within the subsystem use to communicate.

- Cloud Controller Network Communications
- Consul Network Communications
- Diego Network Communications
- Loggregator Network Communications
- MySQL Network Communications
- NATS Network Communications
- Routing Network Communications
- UAA Network Communications

Cloud Controller Network Communications

This topic describes Cloud Controller internal network communication paths with other Pivotal Application Service (PAS) components.

Inbound Communications

The following table lists network communication paths that are inbound to the Cloud Controller.

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
cloud_controller	cloud_controller (Routing API)	443	TCP	HTTPS	OAuth 2.0
diego_brain	cloud_controller	9023	TCP	HTTPS	Mutual TLS
diego_brain (SSH Proxy)	cloud_controller	9022	TCP	HTTP	OAuth 2.0
diego_cell (Rep)	cloud_controller	9023	TCP	HTTPS	Mutual TLS
diego_database (BBS)	cloud_controller	9023	TCP	HTTPS	Mutual TLS
doppler (Syslog Drain Binder)	cloud_controller	9023	TCP	HTTPS	Mutual TLS
loggregator_trafficcontroller	cloud_controller	9023	TCP	HTTPS	Mutual TLS
router	cloud_controller	9022	TCP	HTTP	OAuth 2.0

Outbound Communications

The following table lists network communication paths that are outbound from the Cloud Controller.

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
cloud_controller	mysql_proxy*	3306	TCP	MySQL	MySQL authentication**
cloud_controller	nfs_server or other blobstore†	4443	TCP	HTTPS	TLS and basic authentication
cloud_controller	uaa	8443	TCP	HTTPS	OAuth 2.0 or none‡
cloud_controller	diego_database (BBS)	8889	TCP	HTTPS	Mutual TLS
cloud_controller (Route Registrar)	nats	4222	TCP	NATS	Basic authentication
cloud_controller (Routing API)	diego_database (Locket)	8891	TCP	HTTPS	Mutual TLS
cloud_controller_worker	mysql_proxy*	3306	TCP	MySQL	MySQL authentication**
cloud_controller_worker	nfs_server or other blobstore†	4443	TCP	HTTPS	TLS and basic authentication
clock_global	mysql_proxy*	3306	TCP	MySQL	MySQL authentication**

*Applies only to deployments where internal MySQL is selected as the database.

**MySQL authentication uses the MySQL native password method.

†The destination depends on your file storage or blobstore configuration.

‡The authentication method depends on the type of request.

Consul Communications

PAS components call out to Consul for service discovery. For more information, see Consul Network Communications.

Consul Network Communications

This topic describes Consul internal network communication paths with other Pivotal Application Service (PAS) components.

Consul Communications

The following table lists network communication paths for Consul.

Note: Port 8301 is the destination port for communications between Consul agents. You must allow both TCP and UDP traffic on port 8301 for all VMs running Consul. In addition, `consul_server` communicates with Consul VMs, including other Consul servers, on port 8300.

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
Any VM running Consul	ccdb	8301	TCP and UDP	Gossip (Serf)	Shared secret
Any VM running Consul	clock_global	8301	TCP and UDP	Gossip (Serf)	Shared secret
Any VM running Consul	cloud_controller	8301	TCP and UDP	Gossip (Serf)	Shared secret
Any VM running Consul	cloud_controller_worker	8301	TCP and UDP	Gossip (Serf)	Shared secret
Any VM running Consul	consul_server	8301	TCP and UDP	Gossip (Serf)	Shared secret
Any VM running Consul	diego_brain	8301	TCP and UDP	Gossip (Serf)	Shared secret
Any VM running Consul	diego_cell	8301	TCP and UDP	Gossip (Serf)	Shared secret
Any VM running Consul	diego_database	8301	TCP and UDP	Gossip (Serf)	Shared secret
Any VM running Consul	doppler	8301	TCP and UDP	Gossip (Serf)	Shared secret
Any VM running Consul	ha_proxy	8301	TCP and UDP	Gossip (Serf)	Shared secret
Any VM running Consul	loggregator_trafficcontroller	8301	TCP and UDP	Gossip (Serf)	Shared secret
Any VM running Consul	mysql_proxy*	8301	TCP and UDP	Gossip (Serf)	Shared secret
Any VM running Consul	nats	8301	TCP and UDP	Gossip (Serf)	Shared secret
Any VM running Consul	nfs_server†	8301	TCP and UDP	Gossip (Serf)	Shared secret
Any VM running Consul	router	8301	TCP and UDP	Gossip (Serf)	Shared secret
Any VM running Consul	syslog_adapter	8301	TCP and UDP	Gossip (Serf)	Shared secret
Any VM running Consul	syslog_scheduler	8301	TCP and UDP	Gossip (Serf)	Shared secret
Any VM running Consul	tcp_router	8301	TCP and UDP	Gossip (Serf)	Shared secret
Any VM running Consul	uaa	8301	TCP and UDP	Gossip (Serf)	Shared secret
Any VM running Consul	uaadb	8301	TCP and UDP	Gossip (Serf)	Shared secret
Any VM running Consul	Service tile VMs	8301	TCP and UDP	Gossip (Serf)	Shared secret
Any VM running Consul	consul_server	8300	TCP	HTTPS	Mutual TLS from the same CA

*Applies only to deployments where internal MySQL is selected as the database.

†Applies only to deployments where the internal NFS server is selected for file storage.

Container-to-Container Networking Communications

This topic describes Container-to-Container Networking internal network communication paths with other Pivotal Application Service (PAS) components.

Inbound Communications

The following table lists network communication paths that are inbound to Container-to-Container Networking.

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
diego-cell (Silk CNI)	diego-cell (Silk Daemon)	23954	TCP	HTTP	None
diego-cell (Silk Daemon)	diego-api (Silk Controller)	4103	TCP	HTTP	Mutual TLS
diego-cell (VXLAN Policy Agent)	api (Policy Server Internal)	4003	TCP	HTTP	Mutual TLS

Outbound Communications

The following table lists network communication paths that are outbound from Container-to-Container Networking.

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
api (Policy Server)	uaa	8443	TCP	HTTPS	TLS
api (Policy Server)	api (Cloud Controller)	9022	TCP	HTTP	OAuth 2.0

CredHub Network Communications

This topic describes CredHub internal network communication paths with other Pivotal Application Service (PAS) components.

Inbound Communications

The following table lists network communication paths that are inbound to the CredHub.

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
api	credhub	8844	TCP	HTTPS	OAuth 2.0
diego-cell	credhub	8844	TCP	HTTPS	Mutual TLS†
windows-cell	credhub	8844	TCP	HTTPS	Mutual TLS†
windows2016-cell	credhub	8844	TCP	HTTPS	Mutual TLS†

Outbound Communications

The following table lists network communication paths that are outbound from the CredHub.

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
credhub	uaa	8443	TCP	HTTPS	n/a
credhub	database*	3306	TCP	MySQL	MySQL authentication**

*Applies only to deployments where internal MySQL is selected as the database.

**MySQL authentication uses the MySQL native password method.

†Diego cells use the certificate pairs generated for individual containers to authenticate with CredHub on behalf of applications.

Diego Network Communications

This topic describes Diego internal network communication paths with other Pivotal Application Service (PAS) components.

Inbound Communications

The following table lists network communication paths that are inbound to Diego.

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
cloud_controller	diego_database (BBS)	8889	TCP	HTTPS	Mutual TLS
cloud_controller (Routing API)	diego_database (Locket)	8891	TCP	HTTPS	Mutual TLS

Diego Internal Communications

The following table lists network communication paths that are internal for Diego.

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
diego_brain (Auctioneer)	diego_cell (Rep)	1801	TCP	HTTPS	Mutual TLS
diego_brain (Auctioneer)	diego_database (BBS)	8889	TCP	HTTPS	Mutual TLS
diego_brain (Auctioneer)	diego_database (Locket)	8891	TCP	HTTPS	Mutual TLS
diego_brain (SSH Proxy)	diego_database (BBS)	8889	TCP	HTTPS	Mutual TLS
diego_brain (TPS Watcher)	diego_database (Locket)	8891	TCP	HTTPS	Mutual TLS
diego_cell (local Route Emitter)	diego_database (BBS)	8889	TCP	HTTPS	Mutual TLS
diego_cell (Rep)	diego_brain (CC Uploader)	9090	TCP	HTTP	None
diego_cell (Rep)	diego_brain (File Server)	8080	TCP	HTTP	None
diego_cell (Rep)	diego_database (BBS)	8889	TCP	HTTPS	Mutual TLS
diego_cell (Rep)	diego_database (Locket)	8891	TCP	HTTPS	Mutual TLS
diego_database (BBS)	diego_brain (Auctioneer)	9016	TCP	HTTPS	Mutual TLS
diego_database (BBS)	diego_cell (Rep)	1801	TCP	HTTPS	Mutual TLS
diego_database (BBS)	diego_database (Locket)	8891	TCP	HTTPS	Mutual TLS

Outbound Communications

The following table lists network communication paths that are outbound from Diego.

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
diego_brain	cloud_controller	9023	TCP	HTTPS	Mutual TLS
diego_brain (SSH Proxy)	App instances	2222	TCP	SSH	SSH
diego_brain (SSH Proxy)	cloud_controller	9022	TCP	HTTP	OAuth 2.0
diego_brain (SSH Proxy)	uaa	443	TCP	HTTPS	TLS and OAuth 2.0
diego_cell (local Route Emitter)	nats	4222	TCP	NATS	Basic authentication
diego_cell (Rep)	cloud_controller	9023	TCP	HTTPS	Mutual TLS
diego_cell (Rep)	nfs_server or other blobstore*	Varies	TCP	HTTP	None/TLS
diego_database (BBS)	cloud_controller	9023	TCP	HTTPS	Mutual TLS
diego_database (BBS)	mysql_proxy†	3306	TCP	MySQL	MySQL authentication**
diego_database (Locket)	mysql_proxy†	3306	TCP	MySQL	MySQL authentication**

[*]The destination depends on your PAS blobstore configuration.

[**]MySQL authentication uses the MySQL native password method.

[†]Applies only to deployments where internal MySQL is selected as the database.

Consul Communications

PAS components call out to Consul for service discovery. For more information, see Consul Network Communications.

Loggregator Network Communications

This topic describes Loggregator internal network communication paths with other Pivotal Application Service (PAS) components.

Loggregator Communications

The following table lists network communication paths for Loggregator.

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
Any*	loggregator_trafficcontroller	8081	TCP	HTTP/WebSocket	None
Any VM running Metron	doppler†	8082	TCP	gRPC over HTTP/2	Mutual TLS
loggregator_trafficcontroller	doppler†	8082	TCP	gRPC over HTTP/2	Mutual TLS
loggregator_trafficcontroller	uaa	8443	TCP	HTTPS	TLS
loggregator_trafficcontroller	cloud_controller	9023	TCP	HTTPS	Mutual TLS
loggregator_trafficcontroller (Reverse Log Proxy)	doppler	8082	TCP	gRPC over HTTP/2	Mutual TLS
loggregator_trafficcontroller (Route Registrar)	nats	4222	TCP	NATS	Basic authentication

*Any source VM can send requests to the specified destination within its subnet.

†Starting from ERT v1.11, Metron does not use the UDP protocol to communicate with Doppler. Starting in PAS v2.0, Doppler no longer uses the UDP protocol or the HTTP/WebSocket protocol.

Consul Communications

PAS components call out to Consul for service discovery. For more information, see Consul Network Communications.

MySQL Network Communications

This topic describes MySQL internal network communication paths with other Pivotal Application Service (PAS) components.

These communications only apply to deployments where internal MySQL is selected as the PAS database.

Inbound Communications

The following table lists network communication paths that are inbound to MySQL.

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
cloud_controller	mysql_proxy	3306	TCP	MySQL	MySQL authentication*
cloud_controller_worker	mysql_proxy	3306	TCP	MySQL	MySQL authentication*
clock_global	mysql_proxy	3306	TCP	MySQL	MySQL authentication*
diego_database	mysql_proxy	3306	TCP	MySQL	MySQL authentication*
mysql	mysql (Galera)	4567	TCP	MySQL	MySQL authentication*
mysql_monitor	mysql (Galera health check)	9200	TCP	HTTP	Basic authentication
mysql_monitor	mysql_proxy (Proxy health check)	1936	TCP	MySQL	Basic authentication
mysql_proxy	mysql (MySQL Server)	3306	TCP	MySQL	MySQL authentication*
mysql_proxy	mysql (Proxy API)	8080	TCP	HTTP	Basic authentication
uaa	mysql_proxy	3306	TCP	MySQL	MySQL authentication*

(*) MySQL authentication uses the MySQL native password method.

Outbound Communications

The following table lists network communication paths that are outbound from MySQL.

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
mysql_proxy (Route Registrar)	nats	4222	TCP	NATS	Basic authentication

Consul Communications

All MySQL Proxy instances register themselves with Consul, which enables PAS components that use Consul for service discovery to find available proxies. For more information, see Consul Network Communications.

NATS Network Communications

This topic describes NATS internal network communication paths with other Pivotal Application Service (PAS) components.

Publish Communications

The following table lists network communications that are published to NATS.

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
cloud_controller (Route Registrar)	nats	4222	TCP	NATS	Basic authentication
loggregator_trafficcontroller (Route Registrar)	nats	4222	TCP	NATS	Basic authentication
mysql_proxy (Route Registrar)*	nats	4222	TCP	NATS	Basic authentication
nfs_server (Route Registrar)†	nats	4222	TCP	NATS	Basic authentication
uaa (Route Registrar)	nats	4222	TCP	NATS	Basic authentication
diego_cell (local Route Emitter)	nats	4222	TCP	NATS	Basic authentication

*Applies only to deployments where internal MySQL is selected as the database.

†Applies only to deployments where the internal NFS server is selected for file storage.

Subscribe Communications

The following table lists network communications that are subscribed to NATS.

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
router	nats	4222	TCP	NATS	Basic authentication

Consul Communications

PAS components call out to Consul for service discovery. For more information, see Consul Network Communications.

Routing Network Communications

This topic describes the internal network communication paths of the routing subsystem with other Pivotal Application Service (PAS) components.

HTTP Routing

The following table lists network communication paths for HTTP routing.

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
diego_cell (local Route Emitter)	nats	4222	TCPs	NATS	Basic authentication
Load balancer	router (Gorouter)	80	TCP	HTTP	None
Load balancer	router (Gorouter)	443	TCP	HTTPS	TLS
router (Gorouter)	nats	4222	TCP	NATS	Basic authentication
router (Gorouter)	System components and app containers	Varies	TCP	Varies	None
haproxy	router (Gorouter)	80	TCP	HTTP	None
haproxy	router (Gorouter)	443	TCP	HTTPS	TLS
Load balancer	haproxy	80	TCP	HTTP	None
Load balancer	haproxy	443	TCP	HTTPS	TLS

TCP Routing (Optional)

The following table lists network communication paths for TCP routing.

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
cloud_controller	cloud_controller (Routing API)*	443	TCP	HTTPS	TLS and OAuth 2.0
cloud_controller (Routing API)	diego_database (Locket)	8891	TCP	HTTPS	Mutual TLS
cloud_controller (Routing API)	mysql_proxy	3306	TCP	MySQL	MySQL authentication**
cloud_controller (Routing API)	uaa	8443	TCP	HTTPS	TLS
diego_brain (global TCP Emitter)	cloud_controller (Routing API)	3000	TCP	HTTP	OAuth 2.0
diego_brain (global TCP Emitter)	uaa	8443	TCP	HTTPS	TLS
diego_cell (local Route Emitter)	cloud_controller (Routing API)	3000	TCP	HTTP	OAuth 2.0
diego_cell (local Route Emitter)	uaa	8443	TCP	HTTPS	TLS
Load balancer	tcp_router	1024-65535†	TCP	TCP	None
router (Gorouter)	cloud_controller (Routing API)	3000	TCP	HTTP	OAuth 2.0
router (Gorouter)	uaa	8443	TCP	HTTPS	TLS
tcp_router	cloud_controller (Routing API)	3000	TCP	HTTP	OAuth 2.0
tcp_router	uaa	8443	TCP	HTTPS	TLS

[*] This communication happens through a load balancer and a Gorouter. Requests are received by Routing API on port 3000.

[†] You can use this port range to configure the port in the PAS tile.

[**] MySQL authentication uses the MySQL native password method.

Consul Communications

PAS components call out to Consul for service discovery. For more information, see Consul Network Communications.

Pivotal

UAA Network Communications

This topic describes User Account and Authentication (UAA) internal network communication paths with other Pivotal Application Service (PAS) components.

Inbound Communications

The following table lists network communication paths that are inbound to UAA.

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
cloud_controller	uaa	8443	TCP	HTTPS	OAuth 2.0 or none*
diego_brain (SSH Proxy)	uaa	443	TCP	HTTPS	OAuth 2.0
loggregator_trafficcontroller	uaa	8443	TCP	HTTPS	TLS
router	uaa	8443	TCP	HTTPS	OAuth 2.0

*The authentication method depends on the type of request.

Outbound Communications: Internal to PCF

The following table lists network communication paths that are outbound from UAA.

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
uaa	mysql_proxy*	3306	TCP	MySQL	MySQL authentication**
uaa (Route Registrar)	nats	4222	TCP	NATS	Basic authentication

*Applies only to deployments where internal MySQL is selected as the database.

** MySQL authentication uses the MySQL native password method.

Outbound Communications: External to PCF

The following table lists network communication paths from UAA that are outbound to external systems.

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Authentication
uaa	LDAP	LDAP server communication port	TCP	LDAP/LDAPS	Basic authentication (LDAP bind)
uaa	SAML/OIDC	80 or 443 (HTTP port)	TCP	HTTP/HTTPS	Key

Consul Communications

PAS components call out to Consul for service discovery. For more information, see Consul Network Communications.

CredHub

Overview

CredHub is a component designed for centralized credential management in Pivotal Cloud Foundry (PCF). It is a single component that can address several scenarios in the PCF ecosystem. At the highest level, CredHub centralizes and secures credential generation, storage, lifecycle management, and access.

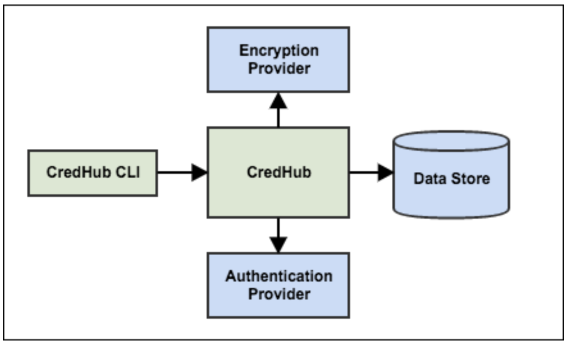
What Can CredHub Do?

CredHub performs a number of different functions to help generate and protect the credentials in your PCF deployment.

- Securing data for storage
- Authentication
- Authorization
- Access and change logging
- Data typing
- Credential generation
- Credential metadata
- Credential versioning

Application Architecture

CredHub consists of a REST API and a CLI. The REST API conforms to the Config Server API spec. CredHub is an OAuth2 resource server that integrates with User Account Authentication (UAA) to provide core authentication and federation capabilities.



CredHub in PCF

A PCF deployment stores credentials in the following locations:

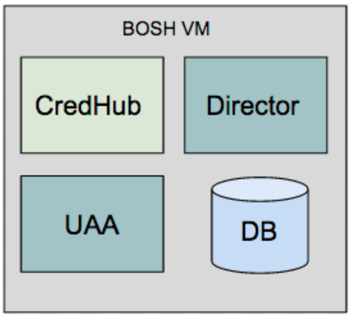
- **BOSH CredHub**: Colocated with the BOSH Director on a single VM. This CredHub instance stores credentials for the BOSH Director.
- **PAS CredHub**: Deployed as an independent service and stores service instance credentials.

BOSH CredHub

In PCF, BOSH Director VM includes a CredHub job. This provides a lightweight credential storage instance for the BOSH Director. The Ops Manager Director, Pivotal Application Service (PAS), and other tiles store credentials in BOSH CredHub. For more information, see Retrieving Credentials from Your Deployment.

Note: This configuration does not provide high availability.

In this colocated deployment architecture, the BOSH Director, CredHub, UAA, and the Director database are all installed on a single BOSH VM, as shown in the following diagram:

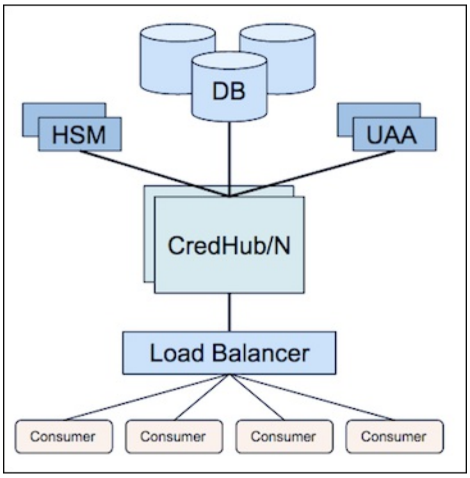


PAS CredHub

The PAS tile deploys CredHub as an independent service on its own VM. This provides a highly available credential storage instance for securing service instance credentials. For more information, see Securing Service Instance Credentials with PAS CredHub.

CredHub is a stateless application, so you can scale it to multiple instances that share a common database cluster and encryption provider.

With CredHub as a service, the load balancer and external databases communicate directly with the CredHub VMs, as shown in the following diagram:



Using CredHub to Store Credentials for Service Tiles

If you develop a service tile for PCF and want to store its credentials in BOSH CredHub, see the CredHub section of the *Tile Developer Guide*.

CredHub Credential Types

Credentials exist in multiple places in the PCF ecosystem. PCF components use credentials to authenticate connections between components. PCF installations often have hundreds of active credentials. Leaked credentials are common causes of data and security breaches, so managing them securely is very important.

For more information, read CredHub Credential Types.

Backing Up and Restoring CredHub Instances

The CredHub application does not hold state, but you must ensure its dependent components are backed up. Redundant backups can help prevent data loss if an individual component fails. For more information, read Backing Up and Restoring CredHub Instances.

CredHub Credential Types

This topic describes the different credential types supported by CredHub.

CredHub supports different types of credentials to simplify generating and managing multi-part credentials. For example, a TLS certificate contains three parts: the root certificate authority (CA), the certificate, and the private key. CredHub supports all three parts, which helps keep connection requests from being rejected erroneously.

CredHub supports the following credential types:

Type	Description
`value`	A single string value for arbitrary configurations and other non-generated or validated strings.
`json`	An arbitrary JSON object for static configurations with many values.
`user`	A single string value for usernames.
`password`	A single string value for passwords and other random string credentials. Values for this type can be automatically generated.
`certificate`	An object containing a root CA, certificate, and private key. Use this type for key pair applications that utilize a certificate, such as TLS connections. Values for this type can be automatically generated.
`rsa`	An object containing an RSA public key and private key without a certificate. Values for this type can be automatically generated.
`ssh`	An object containing an SSH-formatted public key and private key. Values for this type can be automatically generated.

CredHub

CredHub is a secure credential management component that runs on the BOSH VM to minimize the surface area where credentials can be compromised. This topic provides resources for configuring service tiles to use CredHub, instead of encoding credentials in product template and job template files.

See the CredHub documentation for more information.

Overview

Many PCF components use credentials to authenticate connections, and PCF installations often have hundreds of active credentials. Secure credential management is essential to prevent data and security breaches.

In Pivotal Cloud Foundry (PCF) v1.11.0, CredHub runs on the BOSH VM, alongside the BOSH Director and UAA. Ops Manager v1.11 stores its credentials in CredHub, and users can retrieve them using the CredHub API or the **Credentials** tab of the Ops Manager Director tile. Tile developers can embed CredHub calls in manifest snippets and PCF apps can retrieve credentials using the CredHub API.

See Fetching Variable Names and Values for how to fetch variable names and values using the CredHub API.

CredHub Credential Types

CredHub stores and retrieves the following types of credentials:

- `value` — single string value
- `json` — arbitrary JSON object
- `password` — password string
- `certificate` — object containing certificate authority (CA), certificate, and private key
- `ssh` — object containing SSH public key and private key
- `rsa` — object containing RSA public key and private key

For more information, read CredHub Credential Types.

Creating New Variables

To use CredHub in your deployment, you must create new variables and store them in CredHub. By default, variable namespaces are written to prevent collision across deployments, but you can type variable names precisely if you wish.

For more information, read Creating New Variables in CredHub.

Migrating Credentials

To migrate existing non-configurable credentials to CredHub, such as blobstore secrets and backup encryption keys, use the JavaScript migration process. After a successful migration, Ops Manager deletes the migrated credentials from installation.yml.

For more information, read Migrating Existing Credentials to CredHub.

Fetching Variable Names and Values

API endpoints are available to help you find variable names and values for products known to the Ops Manager Director.

For more information, read Fetching Variable Names and Values.

CredHub in Manifest Snippets

Tile developers can embed CredHub in product template and job template manifest snippets using triple-parenthesis notation:

```
manifest: |
  credhub:
    concatenated_password: prefix-((( credhub-password )))-suffix
    password: ((( credhub-password )))
```

PCF v1.11.0 Limitations

PCF v1.11.0 supports CredHub for credential storage, but it does not support the following:

- Automatic backup and restore for CredHub, along with other PCF system components.
- Automatic tile upgrades that migrate all types of credentials defined in property blueprints in previous tile versions, to storage in CredHub.
- Using CredHub to generate new credentials.

Tile authors may choose to wait until PCF supports some or all of these features before incorporating CredHub into their service.

Security-Related PCF Tiles

This section provides information about other services available for Pivotal Cloud Foundry (PCF) deployments.

- Single Sign-On (SSO)

Other Security Topics

This section contains topics related to securing Pivotal Cloud Foundry (PCF) deployments. These topics may come from third party providers, and are not necessarily owned or maintained by PCF. Pivotal provides them for your convenience, but cannot guarantee the accuracy or currency of these documents.

- Security Guidelines for Your IaaS Provider

Security Guidelines for Your IaaS Provider

Pivotal Cloud Foundry supports a variety of Infrastructure as a Service (IaaS) providers. Different IaaS providers require different configuration steps to secure user data, identity information, and credentials.

Security requirements can vary broadly based on the unique configuration and infrastructure of each organization. Rather than provide specific guidance that may not apply to all use cases, Pivotal has collected links to IaaS providers' security and identity management documentation. The documents below may help you understand how your IaaS' security requirements impact your PCF deployment.

Pivotal does not endorse these documents for accuracy or guarantee that their contents apply to all PCF installations.

How to Use This Topic

Find your IaaS provider in the list below. The documentation items linked for each IaaS may help you configure and secure your installation infrastructure.

Amazon Web Services (AWS)

- AWS Identity and Access Management guide
  This guide is a reference for AWS' Identity and Access Management (IAM) features. If you're new to AWS, start here.

- AWS identity documentation
- AWS credential documentation
  This documentation provides a general definition of IAM terms and provide best practices to help you manage IaaS users and permissions.

Google Cloud Platform (GCP)

- GCP authentication documentation
  This developer-facing documentation explains general authentication guidelines for GCP.

Microsoft Azure

- Azure security documentation
  This site has documentation on Azure security tools. It provides a general guide to how to manage IaaS users and credentials.

OpenStack

- OpenStack credential configuration
- OpenStack credential creation
- OpenStack deployment configuration
  These documents provide a general reference for OpenStack service credential management.

VMware vSphere

- vSphere Security guide (PDF)
  This guide contains best practices for securing and managing a vSphere installation.

Buildpacks

Page last updated:

Buildpacks provide framework and runtime support for apps. Buildpacks typically examine your apps to determine what dependencies to download and how to configure the apps to communicate with bound services.

When you push an app, Cloud Foundry automatically detects an appropriate buildpack for it. This buildpack is used to compile or prepare your app for launch.

> **Note**: Cloud Foundry deployments often have limited access to dependencies. This limitation occurs when the deployment is behind a firewall, or when administrators want to use local mirrors and proxies. In these circumstances, Cloud Foundry provides a Buildpack Packager app.

About Buildpacks

For general information about buildpacks, see About Buildpacks.

System Buildpacks

Cloud Foundry includes a set of system buildpacks for common languages and frameworks. This table lists the system buildpacks.

Name	Supported Languages, Frameworks, and Technologies	GitHub Repository
Binary	*n/a*	Binary source
Go	Go	Go source
Java	Grails, Play, Spring, or any other JVM-based language or framework	Java source
.NET Core	.NET Core	.NET Core source
Node.js	Node or JavaScript	Node.js source
PHP	Cake, Symfony, Zend, Nginx, or HTTPD	PHP source
Python	Django or Flask	Python source
Ruby	Ruby, JRuby, Rack, Rails, or Sinatra	Ruby source
Staticfile	HTML, CSS, JavaScript, or Nginx	Staticfile source

Community Buildpacks

You can find a list of unsupported, community-created buildpacks here: cf-docs-contrib.

Customizing and Developing Buildpacks

For information about customizing existing buildpacks and developing new buildpacks, see Customizing and Developing Buildpacks.

For information about updating and releasing a new version of a Cloud Foundry buildpack through the Cloud Foundry Buildpacks Team Concourse pipeline, see Using CI for Buildpacks. You can use this as a model when working with Concourse to build and release new versions of your own buildpacks.

About Buildpacks

Page last updated:

This topic provides links to additional information about using buildpacks. Each of the following are applicable to all supported buildpack languages and frameworks:

- Understanding Buildpacks
- Pushing an Application with Multiple Buildpacks
- Using a Proxy
- Supported Binary Dependencies
- Production Server Configuration

Pivotal

Understanding Buildpacks

Page last updated:

This topic describes how buildpacks work in Cloud Foundry.

Buildpack Scripts

A buildpack repository may contain the following five scripts in the `bin` directory:

- `bin/detect` determines whether or not to apply the buildpack to an app.
- `bin/supply` provides dependencies for an app.
- `bin/finalize` prepares the app for launch.
- `bin/release` provides feedback metadata to Cloud Foundry indicating how the app should be executed.
- `bin/compile` is a deprecated alternative to `bin/supply` and `bin/finalize` .

The `bin/supply` and `bin/finalize` scripts replace the deprecated `bin/compile` script. Older buildpacks may still use `bin/compile` with the latest version of Cloud Foundry. In this case, applying multiple buildpacks to a single app is not supported. Similarly, newer buildpacks may still provide `bin/compile` for compatibility with Heroku and older versions of Cloud Foundry.

The `bin/supply` script is required for non-final buildpacks. The `bin/finalize` (or `bin/compile`) script is required for final buildpacks.

Note: In this document, the terms *non-final buildpack* and *final buildpack*, or *last buildpack*, are used to describe the process of applying multiple buildpacks to an app. See the following example: `cf push APP-NAME -b FIRST-BUILDPACK -b SECOND-BUILDPACK -b FINAL-BUILDPACK` .

Note: If you use only one buildpack for your app, this buildpack behaves as a final, or last, buildpack.

Note: When using multi-buildpack support, the last buildpack in order is the final buildpack, and is able to make changes to the app and determine a start command. All other specified buildpacks are non-final and only supply dependencies.

bin/detect

The `detect` script determines whether or not to apply the buildpack to an app. The script is called with one argument, the `build` directory for the app. The `build` directory contains the app files uploaded when a user performs a `cf push` .

The `detect` script returns an exit code of `0` if the buildpack is compatible with the app. In the case of system buildpacks, the script also prints the buildpack name, version, and other information to `STDOUT` .

The following is an example `detect` script that checks for a Ruby app based on the existence of a `Gemfile` :

```
#!/usr/bin/env ruby

gemfile_path = File.join ARGV[0], "Gemfile"

if File.exist?(gemfile_path)
  puts "Ruby"
  exit 0
else
  exit 1
end
```

Optionally, the buildpack `detect` script can output additional details provided by the buildpack developer. This includes buildpack versioning information and a list of configured frameworks and their associated versions.

The following is an example of the detailed information returned by the Java buildpack:

```
java-buildpack=v3.0-https://github.com/cloudfoundry/java-buildpack.git#3bd15e1 open-jdk-jre=1.8.0_45 spring-auto-reconfiguration=1.7.0_RELEASE tomcat-access-logging-support=2.4.0_RELEASE tomcat-in
```

Note: Cloud Foundry detects only one buildpack by default. When multiple buildpacks are desired, you must explicitly specify them.

For more information, see the Buildpack Detection section below.

bin/supply

The `supply` script provides dependencies for the app and runs for all buildpacks. All output sent to `STDOUT` is relayed to the user through the Cloud Foundry Command Line Interface (cf CLI).

The script is run with four arguments:

- The `build` directory for the app
- The `cache` directory, which is a location the buildpack can use to store assets during the build process
- The `deps` directory, which is where dependencies provided by all buildpacks are installed
- The `index`, which is a number that represents the ordinal position of the buildpack

The `supply` script stores dependencies in `deps` / `index`. It may also look in other directories within `deps` to find dependencies supplied by other buildpacks.

The `supply` script must not modify anything outside of the `deps` / `index` directory. Staging may fail if such modification is detected.

The `cache` directory provided to the `supply` script of the final buildpack is preserved even when the buildpack is upgraded or otherwise changes. The `finalize` script also has access to this cache directory.

The `cache` directories provided to the `supply` scripts of non-final buildpacks are cleared if those buildpacks are upgraded or otherwise change.

The following is an example of a simple `supply` script:

```
#!/usr/bin/env ruby

#sync output

$stdout.sync = true

build_path = ARGV[0]
cache_path = ARGV[1]
deps_path = ARGV[2]
index = ARGV[3]

install_ruby

private

def install_ruby
  puts "Installing Ruby"

  # !!! build tasks go here !!!
  # download ruby
  # install ruby
end
```

bin/finalize

The `finalize` script prepares the app for launch and runs only for the last buildpack. All output sent to `STDOUT` is relayed to the user through the cf CLI.

The script is run with four arguments:

- The `build` directory for the app
- The `cache` directory, which is a location the buildpack can use to store assets during the build process
- The `deps` directory, which is where dependencies provided by all buildpacks are installed
- The `index`, which is a number that represents the ordinal position of the buildpack

The `finalize` script may find dependencies installed by the `supply` script of the same buildpack in `deps` / `index`. It may also look in other directories within `deps` to find dependencies supplied by other buildpacks.

The `cache` directory provided to the `finalize` script is preserved even when the buildpack is upgraded or otherwise changes. The `supply` script of the same buildpack also has access to this cache directory.

The following is an example of a simple `finalize` script:

```
#!/usr/bin/env ruby

#sync output

$stdout.sync = true

build_path = ARGV[0]
cache_path = ARGV[1]
deps_path = ARGV[2]
index = ARGV[3]

setup_ruby

private

def setup_ruby
  puts "Configuring your app to use Ruby"

  # !!! build tasks go here !!!
  # setup ruby
end
```

bin/compile (Deprecated)

The `compile` script is deprecated. It encompasses the behavior of the `supply` and `finalize` scripts for single buildpack apps by using the `build` directory to store dependencies.

The script is run with two arguments:

- The `build` directory for the app
- The `cache` directory, which is a location the buildpack can use to store assets during the build process

During the execution of the `compile` script, all output sent to `STDOUT` is relayed to the user through the cf CLI.

bin/release

The `release` script provides feedback metadata to Cloud Foundry indicating how the app should be executed. The script is run with one argument, the `build` directory. The script must generate a YAML file in the following format:

```
default_process_types:
  web: start_command.filetype
```

`default_process_types` indicates the type of app being run and the command used to start it. This start command is used if a start command is not specified in the `cf push` or in a Procfile.

At this time, only the `web` type of apps is supported.

Note: To define environment variables for your buildpack, add a Bash script to the `.profile.d` directory in the root folder of your app.

The following example shows what a `release` script for a Rack app might return:

```
default_process_types:
  web: bundle exec rackup config.ru -p $PORT
```

Note: The `web` command runs as `bash -c COMMAND` when Cloud Foundry starts your app. Refer to the command attribute section for more information about custom start commands.

Droplet Filesystem

The buildpack staging process extracts the droplet into the `/home/vcap` directory inside the instance container and creates the following filesystem tree:

```
app/
deps/
logs/
tmp/
staging_info.yml
```

The `app` directory includes the contents of the `build` directory, and `staging_info.yml` contains the staging metadata saved in the droplet.

Buildpack Detection

When you push an app, Cloud Foundry uses a detection process to determine a single buildpack to use. For general information about this process, see How Applications Are Staged.

During staging, each buildpack has a position in a priority list. You can retrieve this position by running `cf buildpacks`.

Cloud Foundry checks if the buildpack in position 1 is a compatible buildpack. If the position 1 buildpack is not compatible, Cloud Foundry moves on to the buildpack in position 2. Cloud Foundry continues this process until the correct buildpack is found.

If no buildpack is compatible, the `cf push` command fails with the following error:

```
None of the buildpacks detected a compatible application
Exit status 222
Staging failed: Exited with status 222

FAILED
NoAppDetectedError
```

For a more detailed account of how Cloud Foundry interacts with the buildpack, see the Sequence of Interactions section below.

Sequence of Interactions

This section describes the sequence of interactions between the Cloud Foundry platform and the buildpack. The sequence of interactions differs depending on whether the platform skips or performs buildpack detection.

No Buildpack Detection

Cloud Foundry skips buildpack detection if the developer specifies one or more buildpacks in the app manifest or in the `cf push APP-NAME -b BUILDPACK-NAME` cf CLI command.

If you explicitly specify buildpacks, Cloud Foundry performs the following interactions:

1. For each buildpack except the last buildpack, the platform does the following:
   a. Creates the `deps` / `index` directory
   b. Runs `/bin/supply` with the `build`, `cache`, and `deps` directories and the buildpack `index`
   c. Accepts any modification of the `deps` / `index` directory
   d. Accepts any modification of the `cache` directory
   e. May disallow modification of any other directories
2. For the last buildpack, the platform does the following:
   a. If `/bin/finalize` is present:
      i. Creates the `deps` / `index` directory if it does not exist
      ii. If `/bin/supply` is present, runs `/bin/supply` with the `build`, `cache`, and `deps` directories and the buildpack `index`
      iii. Accepts any modification of the `deps` / `index` directory

iv. May disallow modification of the `build` directory
v. Runs `/bin/finalize` with the `build`, `cache`, and `deps` directories and the buildpack `index`
vi. Accepts any modification of the `build` directory

b. If `/bin/finalize` is not present:

   i. Runs `/bin/compile` with the `build` and `cache` directories
   ii. Accepts any modification of the `build` directory

c. Runs `/bin/release` to determine staging information

At the end of this process, the `deps` directory is included at the root of the droplet, adjacent to the `app` directory.

Buildpack Detection

Cloud Foundry performs buildpack detection if the developer does not specify one or more buildpacks in the app manifest or in the `cf push APP-NAME -b BUILDPACK-NAME` cf CLI command.

> **Note**: Cloud Foundry detects only one buildpack to use with the app.

If the platform performs detection, it does the following:

1. Runs `/bin/detect` for each buildpack

2. Selects the first buildpack with a `/bin/detect` script that returns a zero exit status

3. If `/bin/finalize` is present:

   a. Creates the `deps` / `index` directory if it does not exist
   b. If `/bin/supply` is present, runs `/bin/supply` with the `build`, `cache`, and `deps` directories and the buildpack `index`
   c. Accepts any modification of the `deps` / `index` directory
   d. May disallow modification of the `build` directory
   e. Runs `/bin/finalize` on the `build`, `cache`, and `deps` directories
   f. Accepts any modification of the `build` directory

4. If `/bin/finalize` is not present:

   a. Runs `/bin/compile` on the `build` and `cache` directories
   b. Accepts any modification of the `build` directory

5. Runs `/bin/release` to determine staging information

At the end of this process, the `deps` directory is included at the root of the droplet, adjacent to the `app` directory.

Pushing an Application with Multiple Buildpacks

Page last updated:

This topic describes how developers can push an application with multiple buildpacks.

For more information about pushing applications to Cloud Foundry, see the Deploy an Application topic.

Specifying Buildpacks with the cf CLI

To push an application with multiple buildpacks using the Cloud Foundry Command Line Interface (cf CLI), perform the following procedure:

1. Ensure you are using the cf CLI v6.32.0+:

   ```
   $ cf version
   cf version 6.32.0+0191c33d9.2017-09-26
   ```

   For more information about upgrading the cf CLI, see Installing the cf CLI.

2. Push the application with the binary buildpack with the `--no-start` flag:

   ```
   $ cf push YOUR-APP --no-start -b binary_buildpack
   ```

   This command pushes the application but does not start it.

3. Upgrade the application to multiple buildpacks, and specify the buildpacks:

   ```
   $ cf v3-push YOUR-APP -b BUILDPACK-NAME-1 -b BUILDPACK-NAME-2
   ```

   This command changes the buildpack and starts the application. To see a list of available buildpacks, run `cf buildpacks`.

Note: The two-push workflow is currently required because only `v3-push` supports multiple buildpacks.

For more information about V3 commands, see the Using Experimental cf CLI Commands topic. The `v3-push` command has the following restrictions:

- `v3-push` currently only supports a subset of features of `push`. In particular, it does not support the following:
  - application manifests
  - flags to set the stack or modify the default mapped route
  - exclusions from a `.cfignore` file
- If you use an application manifest, you cannot include the `buildpack` key or future pushes will not function properly.
- You can use the following commands to update the configuration of an application started with `v3-push`:
  - `map-route`
  - `bind-service`
  - `v3-set-env`
  - `v3-scale`
  - `v3-set-health-check`

For more information about V3 commands, see the Using Experimental cf CLI Commands topic.

For more information about using the cf CLI, see the Cloud Foundry Command Line Interface topic.

Using a Proxy

Page last updated:

This topic describes how developers can use a proxy with the buildpacks for their application.

Use a Proxy

Buildpacks can use proxies using the `http_proxy` and `https_proxy` environment variables. You should set these to the proxy hostname or port.

All of the buildpacks automatically use these proxy environment variables correctly. If any buildpacks contacts the Internet during staging, it does so through the proxy host. The binary buildpack does not use a proxy because it does not use the Internet during staging.

To set a proxy for buildpacks to use during staging, perform one of the following procedures:

- Set the environment variables by adding the following section to the `env` block of the application manifest:

```
---
 env:
   http_proxy: http://YOUR-HTTP-PROXY:PORT
   https_proxy: https://YOUR-HTTPS-PROXY:PORT
```

- Set the environment variables with the Cloud Foundry Command Line Interface (cf CLI) using the `cf set-env` command:

```
$ cf set-env YOUR-APP http_proxy "http://YOUR-HTTP-PROXY:PORT"
$ cf set-env YOUR-APP https_proxy "https://YOUR-HTTPs-PROXY:PORT"
```

Note: While many apps use the `http_proxy` and `https_proxy` environment variables at runtime, some do not. The buildpack does not add extra functionality to make proxies work at runtime.

Supported Binary Dependencies

Page last updated:

Each buildpack only supports the stable patches for each dependency listed in the buildpack's `manifest.yml` and also in its GitHub releases page. For example, see the php-buildpack releases page.

If you try to use an unsupported binary, staging your app fails with the following error message:

```
Could not get translated url, exited with: DEPENDENCY_MISSING_IN_MANIFEST:
...
 !
 !     exit
 !
Staging failed: Buildpack compilation step failed
```

Production Server Configuration

Page last updated:

This topic describes how to configure a production server for your apps.

When you deploy an app, PAS determines the command used to start the app through the following process:

1. If the developer uses the command `cf push -c COMMAND`, then PAS uses `COMMAND` to start the app.

2. If the developer creates a file called a Procfile, PAS uses the Procfile to configure the command that launches the app. See the About Procfiles section below for more information.

3. If the developer does not use `cf push -c COMMAND` and does not create a Procfile, then PAS does one of the following, depending on the buildpack:
   - Uses a default start command.
   - Fails to start the app and shows a warning that the app is missing a Procfile.

About Procfiles

One reason to use a Procfile is specify a start command for buildpacks where a default start command is not provided. Some buildpacks, such as Python, that work on a variety of frameworks, do not attempt to provide a default start command.

Another reason to use a Procfile is to configure a production server for web apps.

A Procfile enables you to declare required runtime processes, called process types, for your web app. Process managers in a server use the process types to run and manage the workload. In a Procfile, you declare one process type per line and use the following syntax:

```
PROCESS_TYPE: COMMAND
```

- `PROCESS_TYPE` is `web`. A `web` process handles HTTP traffic.
- `COMMAND` is the command line to launch the process.

For example, a Procfile with the following content starts the launch script created by the build process for a Java app:

```
web: build/install/MY-PROJECT-NAME/bin/MY-PROJECT-NAME
```

Specify a Web Server

Follow these steps to specify a web server using a Procfile. For more information about configuring a web server for Rails apps, see the Configure a Ruby Web Server section of this topic.

1. Create a blank file with a command line for a `web` process type.

2. Save it as a file named `Procfile` with no extension in the root directory of your app.

3. Push your app.

Configure a Ruby Web Server

PAS uses the default standard Ruby web server library WEBrick for Ruby and Ruby on Rails apps. However, PAS can support a more robust production web server, such as Phusion Passenger, Puma, Thin, or Unicorn.

To instruct PAS to use a web server other than WEBrick, perform the following steps:

1. Add the gem for the web server to your Gemfile.

2. In the `config` directory of your app, create a new configuration file or modify an existing file. Refer to your web server documentation for how to configure this file. The following example uses the Puma web server:

```
# config/puma.rb
threads 8,32
workers 3

on_worker_boot do
  # things workers do
end
```

3. In the root directory of your app, create a Procfile and add a command line for a `web` process type that points to your web server.
   For information about configuring the specific command for a process type, see your web server documentation.

   The following example shows a command that starts a Puma web server and specifies the app runtime environment, TCP port, and paths to the server state information and configuration files:

   ```
   web: bundle exec puma -e $RAILS_ENV -p 1234 -S ~/puma -C config/puma.rb
   ```

Pivotal

Binary Buildpack

Page last updated:

Use the binary buildpack for running arbitrary binary web servers.

Push an App

Unlike most other Cloud Foundry buildpacks, you must specify the binary buildpack to use it when staging your binary file. On a command line, use `cf push APP-NAME` with the `-b` option to specify the buildpack.

For example:

```
$ cf push my_app -b https://github.com/cloudfoundry/binary_buildpack.git
```

You can provide Cloud Foundry with the shell command to execute your binary in the following two ways:

- **Procfile**: In the root directory of your app, add a `Procfile` that specifies a `web` task:

```
web: ./app
```

- **Command line**: Use `cf push APP-NAME` with the `-c` option:

```
$ cf push my_app -c './app' -b binary_buildpack
```

Compile your Binary

Cloud Foundry expects your binary to bind to the port specified by the `PORT` environment variable.

The following example in Go binds a binary to the PORT environment variable:

```
package main

import (
    "fmt"
    "net/http"
    "os"
)

func handler(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintf(w, "Hello, %s", "world!")
}

func main() {
    http.HandleFunc("/", handler)
    http.ListenAndServe(":"+os.Getenv("PORT"), nil)
}
```

Your binary should run without any additional runtime dependencies on the cflinuxfs2 or lucid64 root filesystem (rootfs). Any such dependencies should be statically linked to the binary.

To boot a docker container running the cflinuxfs2 filesystem, run the following command:

```
$ docker run -it cloudfoundry/cflinuxfs2 bash
```

To boot a docker container running the lucid64 filesystem, run the following command:

```
$ docker run -it cloudfoundry/lucid64 bash
```

To compile the above Go application on the rootfs, golang must be installed. `apt-get install golang` and `go build app.go` will produce an `app` binary.

When deploying your binary to Cloud Foundry, use `cf push` with the `-s` option to specify the root filesystem it should run against.

```
$ cf push my_app -s (cflinuxfs2|lucid64)
```

To run docker on Mac OS X, we recommend boot2docker.

BOSH Configured Custom Trusted Certificate Support

Your platform operator can configure the platform to add the custom certificates into the application container. The custom trusted certificates will be added to the `/etc/ssl/certs` directory and can be used by binary applications.

Help and Support

Join the #buildpacks channel in our Slack community if you need any further assistance.

For more information about using and extending the binary buildpack in Cloud Foundry, see the binary-buildpack GitHub repository.

You can find current information about this buildpack on the binary buildpack release page in GitHub.

Go Buildpack

Page last updated:

Supported Versions

Supported Go versions can be found in the release notes.

Push an App

The Go buildpack will be automatically detected in the following circumstances:

- Your app has been packaged with godep using `godep save`.
- Your app has a `vendor/` directory and has any files ending with `.go`.
- Your app has a `GOPACKAGENAME` environment variable specified and has any files ending with `.go`.
- Your app has a `glide.yml` file and is using glide, starting in buildpack version 1.7.9.

If your Cloud Foundry deployment does not have the Go Buildpack installed, or the installed version is out of date, you can use the latest version with the command:

```
$ cf push my_app -b https://github.com/cloudfoundry/go-buildpack.git
```

When specifying versions, specify only major/minor versions, such as Go 1.6, rather than Go 1.6.0. This ensures you receive the most recent patches.

Start Command

When pushing Go apps, you can specify a start command for the app. You can place the start command in the `Procfile` file in root directory of your app. For example, if the binary generated by your Go project is `my-go-server`, your `Procfile` could contain the following:

```
web: my-go-server
```

For more information about Procfiles, see the Configuring a Production Server topic.

You can also specify the start command for your app in the `manifest.yml` file in the root directory. For example, your `manifest.yml` could contain the following:

```
---
applications:
  - name: my-app-name
    command: my-go-server
```

If you do not specify a start command in a `Procfile`, in the manifest, or with the `-c` flag for `cf push`, the generated binary will be used as the start command. Example: `my-go-server`

Push an App with godep

If you are using godep to package your dependencies, make sure that you have created a valid `Godeps/Godeps.json` file in the root directory of your app by running `godep save`.

When using godep, you can fix your Go version in `GoVersion` key of the `Godeps/Godeps.json` file.

Go 1.6

- Sample Go 1.6 app

An example `Godeps/Godeps.json` :

```
{
  "ImportPath": "go_app",
  "GoVersion": "go1.6",
  "Deps": []
}
```

Push an App with Glide

If you use glide to specify or package your dependencies, make sure that you have created a valid `glide.yml` file in the root directory of your app by running `glide init` .

To vendor your dependencies before pushing, run `glide install` . This will generate a `vendor` directory and a `glide.lock` file specifying the latest compatible versions of your dependencies. You must have a `glide.lock` file when pushing a vendored app. You do not need a `glide.lock` file when deploying a non-vendored app.

Glide

- Sample Go app with Glide

An example `glide.yml` file:

```
package: go_app_with_glide
import:
- package: github.com/ZiCog/shiny-thing
  subpackages:
  - foo
```

You can specify Go version in the `manifest.yml` file:

```
---
applications:
  - name: my-app-name
    env:
      GOVERSION: go1.8
```

Push an App with Native Go Vendoring

If you use the native Go vendoring system, which packages all local dependencies in the `vendor/` directory, you must specify your app's package name in the `GOPACKAGENAME` environment variable.

An example `manifest.yml` :

```
---
applications:
  - name: my-app-name
    command: go-online
    env:
      GOPACKAGENAME: example.com/user/app-package-name
```

Go 1.6

- Sample Go 1.6 app with native vendoring.

Go 1.6 has vendoring enabled by default. Set the `GO15VENDOREXPERIMENT` environment variable to `0` to disable vendoring.

An example `manifest.yml` file:

```
---
applications:
 - name: my-app-name
   command: example-project
   env:
    GOVERSION: go1.6
    GOPACKAGENAME: example.com/user/app-package-name
```

Go 1.7 and Later

- Sample Go 1.7 app with native vendoring.

Go 1.7 and later always has vendoring enabled, and you cannot disable it with an environment variable.

An example `manifest.yml`:

```
---
applications:
 - name: my-app-name
   command: example-project
   env:
    GOVERSION: go1.8
    GOPACKAGENAME: example.com/user/app-package-name
```

Pass a Symbol and String to the Linker

The Go buildpack supports the Go linker's ability, `-X symbol value`, to set the value of a string at link time. Set the `GO_LINKER_SYMBOL` and `GO_LINKER_VALUE` in the application's configuration before pushing code.

This can be used to embed the commit SHA or other build-specific data directly into the compiled executable.

For example usage, see the relevant fixture app.

C Dependencies

The Go buildpack supports building with C dependencies using cgo. You can set config vars to specify cgo flags to, for example, specify paths for vendored dependencies. As an example, to build gopgsqldriver, add the config var `CGO_CFLAGS` with the value `-I/app/code/vendor/include/postgresql` and include the relevant Postgres header files in `vendor/include/postgresql/` in your app.

Proxy Support

If you need to use a proxy to download dependencies during staging, you can set the `http_proxy` and/or `https_proxy` environment variables. For more information, see the Proxy Usage topic.

BOSH Configured Custom Trusted Certificate Support

Go uses certificates stored in `/etc/ssl/certs`. Your platform operator can configure the platform to add the custom certificates into the application container.

Help and Support

Join the #buildpacks channel in our Slack community if you need any further assistance.

For more information about using and extending the Go buildpack in Cloud Foundry, see the go-buildpack GitHub repository.

You can find current information about this buildpack on the Go buildpack release page in GitHub.

Java Buildpack

You can use use the Java buildpack with apps written in Grails, Play, Spring, or any other JVM-based language or framework.

See the following topics for more information:

- Tips for Java Developers
- Getting Started Deploying Apps
- Configuring Service Connections
- Cloud Foundry Eclipse Plugin
- Cloud Foundry Java Client Library

See the Java Buildpack Release Notes for information about specific versions. You can find the source for the Java buildpack in the Java buildpack repository on GitHub:

Buildpack and Application Logging

The Java buildpack only runs during the staging process, and therefore only logs staging information such as the downloaded components, configuration data, and work performed on your application by the buildpack.

The Java buildpack source documentation states the following:

- The Java buildpack logs all messages, regardless of severity, to `APP-DIRECTORY/.java-buildpack.log`. The buildpack also logs messages to `$stderr`, filtered by a configured severity level.
- If the buildpack fails with an exception, the exception message is logged with a log level of `ERROR`. The exception stack trace is logged with a log level of `DEBUG`. This prevents users from seeing stack traces by default.

Once staging completes, the buildpack stops logging. The Loggregator handles application logging.

Your application must write to STDOUT or STDERR for its logs to be included in the Loggregator stream. For more information, see the Application Logging in Cloud Foundry topic.

BOSH Custom Trusted Certificate Support

Versions 3.7 and later of the Java buildpack support BOSH-configured custom trusted certificates. For more information, see Configuring Trusted Certificates in the BOSH documentation.

The Java buildpack pulls the contents of `/etc/ssl/certs/ca-certificates.crt` and `$CF_INSTANCE_CERT/$CF_INSTANCE_KEY` by default.

The log output for Diego Instance Identity-based `KeyStore` appears as follows:

```
Adding System Key Manager
Adding Key Manager for /etc/cf-instance-credentials/instance.key and /etc/cf-instance-credentials/instance.crt
Start watching /etc/cf-instance-credentials/instance.crt
Start watching /etc/cf-instance-credentials/instance.key
Initialized KeyManager for /etc/cf-instance-credentials/instance.key and /etc/cf-instance-credentials/instance.crt
```

The log output for Diego Trusted Certificate-based `TrustStore` appears as follows:

```
Adding System Trust Manager
Adding TrustManager for /etc/ssl/certs/ca-certificates.crt
Start watching /etc/ssl/certs/ca-certificates.crt
Initialized TrustManager for /etc/ssl/certs/ca-certificates.crt
```

Memory Constraints in Java Buildpack 4.0

The memory calculator in Java buildpack 4.0 accounts for the following memory regions:

- `-Xmx` : Heap

- `-XX:MaxMetaspaceSize` : Metaspace
- `-Xss` : Thread Stacks
- `-XX:MaxDirectMemorySize` : Direct Memory
- `-XX:ReservedCodeCacheSize` : Code Cache
- `-XX:CompressedClassSpaceSize` : Compressed Class Space

Applications which previously ran in 512 MB or smaller containers may no longer be able to. Most applications will run if they use the Cloud Foundry default container size of 1 G without any modifications. However, you can configure those memory regions directly as needed.

The Java buildpack optimizes for all non-heap memory regions first and leaves the remainder for the heap.

The Java buildpack prints a histogram of the heap to the logs when the JVM encounters a terminal failure.

The Cloud Foundry default Java buildpack is currently 3.x to allows time for apps to be upgrade to 4.x.

For more information, see Java buildpack 4.0.

Tips for Java Developers

Page last updated:

Cloud Foundry can deploy a number of different JVM-based artifact types. For a more detailed explanation of what it supports, see the Java Buildpack documentation.

Java Buildpack

For detailed information about using, configuring, and extending the Cloud Foundry Java buildpack, see Java Buildpack documentation.

Design

The Java Buildpack is designed to convert artifacts that run on the JVM into executable applications. It does this by identifying one of the supported artifact types (Grails, Groovy, Java, Play Framework, Spring Boot, and Servlet) and downloading all additional dependencies needed to run. The collection of services bound to the application is also analyzed and any dependencies related to those services are also downloaded.

As an example, pushing a WAR file that is bound to a PostgreSQL database and New Relic for performance monitoring would result in the following:

```
Initialized empty Git repository in /tmp/buildpacks/java-buildpack/.git/
--> Java Buildpack source: https://github.com/cloudfoundry/java-buildpack#0928916a2dd78e9faf9469c558046eef09f60e5d
--> Downloading Open Jdk JRE 1.7.0_51 from
    http://.../openjdk/lucid/x86_64/openjdk-1.7.0_51.tar.gz (0.0s)
     Expanding Open Jdk JRE to .java-buildpack/open_jdk_jre (1.9s)
--> Downloading New Relic Agent 3.4.1 from
    http://.../new-relic/new-relic-3.4.1.jar (0.4s)
--> Downloading Postgresql JDBC 9.3.1100 from
    http://.../postgresql-jdbc/postgresql-jdbc-9.3.1100.jar (0.0s)
--> Downloading Spring Auto Reconfiguration 0.8.7 from
    http://.../auto-reconfiguration/auto-reconfiguration-0.8.7.jar (0.0s)
     Modifying /WEB-INF/web.xml for Auto Reconfiguration
--> Downloading Tomcat 7.0.50 from
    http://.../tomcat/tomcat-7.0.50.tar.gz (0.0s)
     Expanding Tomcat to .java-buildpack/tomcat (0.1s)
--> Downloading Buildpack Tomcat Support 1.1.1 from
    http://.../tomcat-buildpack-support/tomcat-buildpack-support-1.1.1.jar (0.1s)
--> Uploading droplet (57M)
```

Configuration

In most cases, the buildpack should work without any configuration. If you are new to Cloud Foundry, we recommend that you make your first attempts without modifying the buildpack configuration. If the buildpack requires some configuration, use a fork of the buildpack.

Java Client Library

The Cloud Foundry Client Library provides a Java API for interacting with a Cloud Foundry instance. This library, `cloudfoundry-client-lib`, is used by the Cloud Foundry Maven plugin, the Cloud Foundry Gradle plugin, the Cloud Foundry STS integration, and other Java-based tools.

For information about using this library, see the Java Cloud Foundry Library page.

Grails

Grails packages applications into WAR files for deployment into a Servlet container. To build the WAR file and deploy it, run the following:

```
$ grails prod war
$ cf push my-application -p target/my-application-version.war
```

Groovy

Groovy applications based on both Ratpack and a simple collection of files are supported.

Ratpack

Ratpack packages applications into two different styles; Cloud Foundry supports the `distZip` style. To build the ZIP and deploy it, run the following:

```
$ gradle distZip
$ cf push my-application -p build/distributions/my-application.zip
```

Raw Groovy

Groovy applications that are made up of a single entry point plus any supporting files can be run without any other work. To deploy them, run the following:

```
$ cf push my-application
```

Java Main

Java applications with a `main()` method can be run provided that they are packaged as self-executable JARs.

Note: If your application is not web-enabled, you must suppress route creation to avoid a “failed to start accepting connections” error. To suppress route creation, add `no-route: true` to the application manifest or use the `--no-route` flag with the `cf push` command.

For more information about the `no-route` attribute, see the Deploying with Application Manifests topic.

Maven

A Maven build can create a self-executable JAR. To build and deploy the JAR, run the following:

```
$ mvn package
$ cf push my-application -p target/my-application-version.jar
```

Gradle

A Gradle build can create a self-executable JAR. To build and deploy the JAR, run the following:

```
$ gradle build
$ cf push my-application -p build/libs/my-application-version.jar
```

Play Framework

The Play Framework packages applications into two different styles. Cloud Foundry supports both the `staged` and `dist` styles. To build the `dist` style and deploy it, run the following:

```
$ play dist
$ cf push my-application -p target/universal/my-application-version.zip
```

Spring Boot CLI

Spring Boot can run applications comprised entirely of POGOs. To deploy then, run the following:

```
$ spring grab *.groovy
$ cf push my-application
```

Servlet

Java applications can be packaged as Servlet applications.

Maven

A Maven build can create a Servlet WAR. To build and deploy the WAR, run the following:

```
$ mvn package
$ cf push my-application -p target/my-application-version.war
```

Gradle

A Gradle build can create a Servlet WAR. To build and deploy the JAR, run the following:

```
$ gradle build
$ cf push my-application -p build/libs/my-application-version.war
```

Binding to Services

Information about binding apps to services can be found on the following pages:

- Service Bindings for Grails Applications
- Service Bindings for Play Framework Applications
- Service Bindings for Spring Applications

Java and Grails Best Practices

Provide JDBC driver

The Java buildpack does not bundle a JDBC driver with your application. If your application will access a SQL RDBMS, include the appropriate driver in your application.

Allocate Sufficient Memory

If you do not allocate sufficient memory to a Java application when you deploy it, it may fail to start, or PAS may terminate it. You must allocate enough memory to allow for the following:

- Java heap
- Metaspace, if using Java 8
- PermGen, if using Java 7 or earlier
- Stack size per Thread

- JVM overhead

The `config/open_jdk_jre.yml` file of the Cloud Foundry Java buildpack contains default memory size and weighting settings for the JRE. See the Open JDK JRE README on GitHub for an explanation of JRE memory sizes and weightings and how the Java buildpack calculates and allocates memory to the JRE for your app.

To configure memory-related JRE options for your app, you either create a custom buildpack and specify this buildpack in your deployment manifest or you override the default memory settings of your buildpack as described in the https://github.com/cloudfoundry/java-buildpack#configuration-and-extension with the properties listed in the Open JDK JRE README. For more information about configuring custom buildpacks and manifests, refer to the Custom Buildpacks and Deploying with Application Manifests topics.

When your app is running, you can use the `cf app APP-NAME` command to see memory utilization.

Troubleshoot Out Of Memory

A Java app may crash because of insufficient memory on the Garden container or the JVM on which it runs. See the following sections for help diagnosing and resolving such issues.

JVM

- **Error**: `java.lang.OutOfMemoryError`. See the following example:

```
$ cf logs APP-NAME --recent
2016-06-20T09:18:51.00+0100 [APP/0] OUT java.lang.OutOfMemoryError: Metaspace
```

- **Cause**: If the JVM cannot garbage-collect enough space to ensure the allocation of a data-structure, it fails with java.lang.OutOfMemoryError. In the example above, JVM has an under-sized `metaspace`. You may see failures in other memory pools, such as `heap`.
- **Solution**: Configure the JVM correctly for your app. See Allocate Sufficient Memory.

Garden Container

> **Note**: The solutions in this section require configuring the memory calculator, which is a sub-project of the CF Java buildpack that calculates suitable memory settings for Java apps when you push them. See the java-buildpack-memory-calculator repository for more information. If you have questions about the memory calculator, you can ask them in the **#java-buildpack** channel of the Cloud Foundry Slack organization.

- **Error**: The Garden container terminates the Java process with the `out of memory` event. See the following example:

```
$ cf events APP-NAME
time                          event       actor       description
2016-06-20T09:18:51.00+0100   app.crash   app-name    index: 0, reason: CRASHED, exit_description: out of memory, exit_status: 255
```

  This error appears when the JVM allocates more OS-level memory than the quota requested by the app, such as through the manifest.

- **Cause 1 - Insufficient native memory**: This error commonly means that the JVM requires more `native` memory. In the scope of the Java buildpack and the memory calculator, the term `native` means the memory required for the JVM to work, along with forms of memory not covered in the other classifications of the memory calculator. This includes the memory footprint of OS-level threads, direct NIO buffers, code cache, program counters, and others.
- **Solution 1**: Determine how much `native` memory a Java app needs by measuring it with realistic workloads and fine-tuning it accordingly. You can then configure the Java buildpack using the `native` setting of the memory calculator, as in the example below:

```
---
applications:
- name: app-name
  memory: 1G
  env:
    JBP_CONFIG_OPEN_JDK_JRE: '[memory_calculator: {memory_sizes: {native: 150m}}]'
```

  This example shows that `150m` of the overall available `1G` is reserved for anything that is not `heap`, `metaspace`, or `permgen`. In less common cases, this may come from companion processes started by the JVM, such as the Process API.

- **Cause 2 - High thread count**: Java threads in the JVM can cause memory errors at the Garden level. When an app is under heavy load, it uses a high number of threads and consumes a significant amount of memory.

- **Solution 2**: Set the reserved memory for stack traces to the correct value for your app.

  You can use the `stack` setting of the memory calculator to configure the amount of space the JVM reserves for each Java thread. You must multiply this value by the number of threads your app requires. Specify the number of threads in the `stack_threads` setting of the memory calculator. For example, if you estimate the max thread count for an app at `800` and the amount of memory needed to represent the deepest stacktrace of a Java thread is `512KB`, configure the memory calculator as follows:

  ```
  ---
  applications:
  - name: app-name
    memory: 1G
    env:
      JBP_CONFIG_OPEN_JDK_JRE: '[memory_calculator: {stack_threads: 800, memory_sizes: {stack: 512k}}]'
  ```

  In this example, the overall memory amount reserved by the JVM for representing the stacks of Java threads is `800 * 512k = 400m`.

  The correct settings for `stack` and `stack_threads` depend on your app code, including the libraries it uses. Your app may technically have no upper limit, such as in the case of cavalier usage of `CachedThreadPool` executors. However, you still must calculate the depth of the thread stacks and the amount of space the JVM should reserve for each of them.

Troubleshoot Failed Upload

If your application fails to upload when you push it to Cloud Foundry, it may be for one of the following reasons:

- **WAR is too large**: An upload may fail due to the size of the WAR file. Cloud Foundry testing indicates WAR files as large as 250 MB upload successfully. If a WAR file larger than that fails to upload, it may be a result of the file size.
- **Connection issues**: Application uploads can fail if you have a slow Internet connection, or if you upload from a location that is very remote from the target Cloud Foundry instance. If an application upload takes a long time, your authorization token can expire before the upload completes. A workaround is to copy the WAR to a server that is closer to the Cloud Foundry instance, and push it from there.
- **Out-of-date cf CLI client**: Upload of a large WAR is faster and hence less likely to fail if you are using a recent version of the cf CLI. If you are using an older version of the cf CLI client to upload a large WAR, and having problems, try updating to the latest version of the cf CLI.
- **Incorrect WAR targeting**: By default, `cf push` uploads everything in the current directory. For a Java application, `cf push` with no option flags uploads source code and other unnecessary files, in addition to the WAR. When you push a Java application, specify the path to the WAR:

  ```
  $ cf push MY-APP -p PATH/TO/WAR-FILE
  ```

  You can determine whether or not the path was specified for a previously pushed application by examining the application deployment manifest, `manifest.yml`. If the `path` attribute specifies the current directory, the manifest includes a line like the following:

  ```
  path: .
  ```

  To re-push just the WAR, do one of the following:

  - Delete `manifest.yml` and run `cf push` again, specifying the location of the WAR using the `-p` flag.
  - Edit the `path` argument in `manifest.yml` to point to the WAR, and re-push the application.

Debug Java Apps on Cloud Foundry

Because of the way that Cloud Foundry deploys your applications and isolates them, it is not possible to connect to your application with the remote Java debugger. Instead, instruct the application to connect to the Java debugger on your local machine.

Here are the instructions for setting up remote debugging when using BOSH Lite or a CloudFoundry installation.

1. Open your project in Eclipse.
2. Right-click on your project, go to **Debug as** and pick **Debug Configurations**.
3. Create a new **Remote Java Application**.
4. Make sure your project is selected, pick **Standard (Socket Listen)** from the **Connection Type** drop down and set a port. Make sure this port is open if you are running a firewall.
5. Click **Debug**.

The debugger should now be running. If you switch to the Debug perspective, you should see your application listed in the Debug panel and it should say `Waiting for vm to connect at port`.

Next, push your application to Cloud Foundry and instruct Cloud Foundry to connect to the debugger running on your local machine using the following instructions:

1. Edit your `manifest.yml` file. Set the instances count to 1. If you set this greater than one, multiple applications try to connect to your debugger.
2. Also in `manifest.yml`, add the `env` section and create a variable called `JAVA_OPTS`.
3. Add the remote debugger configuration to the `JAVA_OPTS` variable: `-agentlib:jdwp=transport=dt_socket,address=YOUR-IP-ADDRESS:YOUR-PORT`.
4. Save the `manifest.yml` file.
5. Run `cf push`.

Upon completion, you should see that your application has started and is now connected to the debugger running in your IDE. You can now add breakpoints and interrogate the application just as you would if it were running locally.

Slow Starting Java or Grails Apps

Some Java and Grails applications do not start quickly, and the health check for an application can fail if an application starts too slowly.

The current Java buildpack implementation sets the Tomcat `bindOnInit` property to `false`. This prevents Tomcat from listening for HTTP requests until an application has fully deployed.

If your application does not start quickly, the health check may fail because it checks the health of the application before the application can accept requests. By default, the health check fails after a timeout threshold of 60 seconds.

To resolve this issue, use `cf push APP-NAME` with the `-t TIMEOUT-THRESHOLD` option to increase the timeout threshold. Specify `TIMEOUT-THRESHOLD` in seconds.

```
$ cf push my-app -t 180
```

Note: The timeout threshold cannot exceed 180 seconds. Specifying a timeout threshold greater than 180 seconds results in the following error: `Server error, status code: 400, error code: 100001, message: The app is invalid: health_check_timeout maximum_exceeded`

Extension

The Java Buildpack is also designed to be easily extended. It creates abstractions for three types of components (containers, frameworks, and JREs) in order to allow users to easily add functionality. In addition to these abstractions, there are a number of utility classes for simplifying typical buildpack behaviors.

As an example, the New Relic framework looks like the following:

```
class NewRelicAgent < JavaBuildpack::Component::VersionedDependencyComponent

  # @macro base_component_compile
  def compile
    FileUtils.mkdir_p logs_dir

    download_jar
    @droplet.copy_resources
  end

  # @macro base_component_release
  def release
    @droplet.java_opts
    .add_javaagent(@droplet.sandbox + jar_name)
    .add_system_property('newrelic.home', @droplet.sandbox)
    .add_system_property('newrelic.config.license_key', license_key)
    .add_system_property('newrelic.config.app_name', "'#{application_name}'")
    .add_system_property('newrelic.config.log_file_path', logs_dir)
  end

  protected

  # @macro versioned_dependency_component_supports
  def supports?
    @application.services.one_service? FILTER, 'licenseKey'
  end

  private

  FILTER = /newrelic/.freeze

  def application_name
    @application.details['application_name']
  end

  def license_key
    @application.services.find_service(FILTER)['credentials']['licenseKey']
  end

  def logs_dir
    @droplet.sandbox + 'logs'
  end

end
```

Environment Variables

You can access environments variable programmatically.

For example, you can obtain `VCAP_SERVICES` as follows:

```
System.getenv("VCAP_SERVICES");
```

See the Cloud Foundry Environment Variables topic for more information.

Getting Started Deploying Java Apps

Page last updated:

This topic provides links to additional information about getting started deploying apps using the Java buildpack. See the following topics for deployment guides specific to your app framework:

- Grails
- Ratpack
- Spring

Getting Started Deploying Grails Apps

Page last updated:

This guide is intended to walk you through deploying a Grails app to Pivotal Application Service. If you experience a problem following the steps below, refer to the Known Issues or Troubleshooting Application Deployment and Health topics for more information.

Sample App Step
If you want to go through this tutorial using the sample app, run `git clone https://github.com/cloudfoundry-samples/pong_matcher_grails.git` to clone the `pong_matcher_grails` app from GitHub, and follow the instructions in the Sample App Step sections.

Note: Ensure that your Grails app runs locally before continuing with this procedure.

Deploy a Grails Application

This section describes how to deploy a Grails application to PAS.

Prerequisites

- A Grails app that runs locally on your workstation
- Intermediate to advanced Grails knowledge
- The Cloud Foundry Command Line Interface (cf CLI)
- JDK 1.7 or 1.8 for Java 7 or 8 configured on your workstation

Note: You can develop Grails applications in Groovy, Java 7 or 8, or any JVM language. The Cloud Foundry Java buildpack uses JDK 1.8, but you can modify the buildpack and the manifest for your app to compile to JDK 1.7 as described in *Step 8: Configure the Deployment Manifest* of this topic.

Step 1: Declare App Dependencies

Declare all the dependency tasks for your app in the build script of your chosen build tool. The table lists build script information for Gradle, Grails, and Maven, and provides documentation links for each build tool.

Build Tool	Build Script	Documentation
Gradle	`build.gradle`	Gradle User Guide
Grails	`BuildConfig.groovy`	Grails: Configuration - Reference Documentation
Maven	`pom.xml`	Apache Maven Project Documentation

Sample App Step
You can skip this step. The `pong_matcher_grails/app/grails-app/conf/BuildConfig.groovy` file contains the dependencies for the `pong_matcher_grails` sample app, as the example below shows.

```
dependencies {
      // specify dependencies here under either 'build', 'compile', 'runtime', 'test' or 'provided' scopes e.g.
      // runtime 'mysql:mysql-connector-java:5.1.29'
      // runtime 'org.postgresql:postgresql:9.3-1101-jdbc41'
      test "org.grails:grails-datastore-test-support:1.0-grails-2.4"
      runtime 'mysql:mysql-connector-java:5.1.33'
  }
```

Step 2: Allocate Sufficient Memory

Run the Cloud Foundry Command Line Interface (cf CLI) `cf push -m` command to specify the amount of memory that should be allocated to the application. Memory allocated this way is done in preset amounts of `64M`, `128M`, `256M`, `512M`, `1G`, or `2G`. For example:

```
$ cf push -m 128M
```

When your app is running, you can use the `cf app APP-NAME` command to see memory utilization.

Sample App Step
You can skip this step. In the `manifest.yml` of the `pong_matcher_grails` sample app, the `memory` sub-block of the `applications` block allocates 1 GB to the app.

Step 3: Provide a JDBC Driver

The Java buildpack does not bundle a JDBC driver with your application. If your application accesses a SQL RDBMS, you must do the following:

- Include the appropriate driver in your application.
- Create a dependency task for the driver in the build script for your build tool or IDE.

Sample App Step
You can skip this step. The `pong_matcher_grails` sample app declares a MySQL JDBC driver in the `pong_matcher_grails/app/grails-app/conf/DataSource.groovy` file because the app uses ClearDB, which is a database-as-service for MySQL-powered apps. The example below shows this declaration.

```
dataSource {
  pooled = true
  jmxExport = true
  driverClassName = "com.mysql.jdbc.Driver"
  dialect = org.hibernate.dialect.MySQL5InnoDBDialect
  uri = new URI(System.env.DATABASE_URL ?: "mysql://foo:bar@localhost")
  username = uri.userInfo ? uri.userInfo.split(":")[0] : ""
  password = uri.userInfo ? uri.userInfo.split(":")[1] : ""
  url = "jdbc:mysql://" + uri.host + uri.path

  properties {
    dbProperties {
      autoReconnect = true
    }
  }
}
```

Step 4: (Optional) Configure a Procfile

Use a Procfile to declare required runtime processes for your web app and to specify your web server. For more information, see the Configuring a Production Server topic.

Sample App Step
You can skip this step. The `pong_matcher_grails` app does not require a Procfile.

Step 5: Create and Bind a Service Instance for a Grails Application

This section describes using the cf CLI to configure a ClearDB managed service instance for an app. You can use either the CLI or Apps Manager to perform this task.

PAS supports two types of service instances:

- Managed services integrate with PAS through service brokers that offer services and plans and manage the service calls between PAS and a service provider.
- User-provided service instances enable you to connect your application to pre-provisioned external service instances.

For more information about creating and using service instances, refer to the Services Overview topic.

Create a Service Instance

Run the cf CLI `cf marketplace` command to view managed and user-provided services and plans available to you.

The example shows two of the available managed database-as-a-service providers and their offered plans: `cleardb` database-as-a-service for MySQL-powered apps and `elephantsql` PostgreSQL as a Service.

```
$ cf marketplace
Getting services from marketplace in org Cloud-Apps / space development as clouduser@example.com...
OK

service      plans                    description

cleardb      spark, boost, amp        Highly available MySQL for your apps
elephantsql  turtle, panda, elephant  PostgreSQL as a Service
```

Run `cf create-service SERVICE PLAN SERVICE-INSTANCE` to create a service instance for your app. Choose a SERVICE and PLAN from the list, and provide a unique name for the SERVICE-INSTANCE.

Sample App Step
Run `cf create-service cleardb spark mysql`. This creates a service instance named `mysql` that uses the `cleardb` service and the `spark` plan, as the example below shows.

```
$ cf create-service cleardb spark mysql
Creating service mysql in org Cloud-Apps / space development as clouduser@example.com....
OK
```

Bind a Service Instance

When you bind an app to a service instance, PAS writes information about the service instance to the VCAP_SERVICES app environment variable. The app can use this information to integrate with the service instance.

Most services support bindable service instances. Refer to your service provider's documentation to confirm if they support this functionality.

You can bind a service to an application with the command `cf bind-service APPLICATION SERVICE-INSTANCE`.

Alternately, you can configure the deployment manifest file by adding a `services` sub-block to the `applications` block and specifying the service instance. For more information and an example on service binding using a manifest, see the Sample App step.

You can also bind a service using the Apps Manager.

Sample App Step
You can skip this step because the service instance is already bound. Open the `manifest.yml` file in a text editor to view the bound service instance information. Locate the file in the app root directory and search for the `services` sub-block in the `applications` block, as the example below shows.

```
---
applications:
```

```
...
  services:
    - mysql
```

Step 6: Configure the Deployment Manifest

You can specify deployment options in the `manifest.yml` that the `cf push` command uses when deploying your app.

Refer to the Deploying with Application Manifests topic for more information.

Sample App Step
You can skip this step. The `manifest.yml` file for the `pong_matcher_grails` sample app does not require any additional configuration to deploy the app.

Step 7: Log in and Target the API Endpoint

Run `cf login -a API-ENDPOINT`, enter your login credentials, and select a space and org. The API endpoint is the URL of the Cloud Controller in your PAS instance.

Sample App Step
You must do this step to run the sample app.

Step 8: Deploy the Application

Note: You must use the cf CLI to deploy apps.

From the root directory of your application, run `cf push APP-NAME -p PATH-TO-FILE.war` to deploy your application.

Note: You must deploy the `.war` artifact for a Grails app, and you must include the path to the `.war` file in the `cf push` command using the `-p` option if you do not declare the path in the `applications` block of the manifest file. For more information, refer to the Grails section in the Tips for Java Developers topic.

`cf push APP-NAME` creates a URL route to your application in the form HOST.DOMAIN, where HOST is your APP-NAME and DOMAIN is specified by your administrator. Your DOMAIN is `shared-domain.example.com`. For example: `cf push my-app` creates the URL `my-app.shared-domain.example.com`.

The URL for your app must be unique from other apps that PAS hosts or the push will fail. Use the following options to help create a unique URL:

- `-n` to assign a different HOST name for the app
- `--random-route` to create a URL that includes the app name and random words
- `cf help push` to view other options for this command

If you want to view log activity while the app deploys, launch a new terminal window and run `cf logs APP-NAME`.

Once your app deploys, browse to your app URL. Search for the `urls` field in the `App started` block in the output of the `cf push` command. Use the URL to access your app online.

Sample App Step
1. Change to the `app` directory, and run `./grailsw war` to build the app.
2. Run `cf push pong_matcher_grails -n HOST-NAME` to push the app.

Example: `cf push pong_matcher_grails -n my-grails-app`

Note: You do not have to include the `-p` flag when you deploy the sample app. The sample app manifest declares the path to the archive that `cf push` uses to upload the app files.

The example below shows the terminal output of deploying the `pong_matcher_grails` app. `cf push` uses the instructions in the manifest file to create the app, create and bind the route, and upload the app. It then binds the app to the `mysql` service and follows the instructions in the manifest to start two instances of the app, allocating 1 GB of memory between the instances. After the instances start, the output displays their health and status.

```
$ cf push pong_matcher_grails -n my-grails-app
Using manifest file /Users/example/workspace/pong_matcher_grails/app/manifest.yml

Creating app pong_matcher_grails in org Cloud-Apps / space development as clouduser@example.com...
OK

Creating route my-grails-app.cfapps.io...
OK

Binding my-grails-app.cfapps.io to pong_matcher_grails...
OK

Uploading pong_matcher_grails...
Uploading app files from: /Users/example/workspace/pong_matcher_grails/app/target/pong_matcher_grails-0.1.war
Uploading 4.8M, 704 files
OK
Binding service mysql to app pong_matcher_grails in org Cloud-Apps / space development as clouduser@example.com...
OK

Starting app pong_matcher_grails in org Cloud-Apps / space development as clouduser@example.com...
OK
-----> Downloaded app package (38M)
-----> Java Buildpack Version: v2.5 | https://github.com/cloudfoundry/java-buildpack.git#840500e
-----> Downloading Open Jdk JRE 1.8.0_25 from https://download.run.pivotal.io/openjdk/lucid/x86_64/openjdk-1.8.0_25.tar.gz (1.5s)
       Expanding Open Jdk JRE to .java-buildpack/open_jdk_jre (1.1s)
-----> Downloading Spring Auto Reconfiguration 1.5.0_RELEASE from https://download.run.pivotal.io/auto-reconfiguration/auto-reconfiguration-1.5.0_RELEASE.jar (0.0s)
       Modifying /WEB-INF/web.xml for Auto Reconfiguration
-----> Downloading Tomcat Instance 8.0.14 from https://download.run.pivotal.io/tomcat/tomcat-8.0.14.tar.gz (0.4s)
       Expanding Tomcat to .java-buildpack/tomcat (0.1s)
-----> Downloading Tomcat Lifecycle Support 2.4.0_RELEASE from https://download.run.pivotal.io/tomcat-lifecycle-support/tomcat-lifecycle-support-2.4.0_RELEASE.jar (0.0s)
-----> Downloading Tomcat Logging Support 2.4.0_RELEASE from https://download.run.pivotal.io/tomcat-logging-support/tomcat-logging-support-2.4.0_RELEASE.jar (0.0s)
-----> Downloading Tomcat Access Logging Support 2.4.0_RELEASE from https://download.run.pivotal.io/tomcat-access-logging-support/tomcat-access-logging-support-2.4.0_RELEASE.jar (

-----> Uploading droplet (83M)

0 of 2 instances running, 2 starting
0 of 2 instances running, 2 starting
0 of 2 instances running, 2 starting
2 of 2 instances running

App started

Showing health and status for app pong_matcher_grails in org Cloud-Apps / space development as clouduser@example.com...
OK

requested state: started
instances: 2/2
usage: 1G x 2 instances
urls: my-grails-app.cfapps.io

     state     since                    cpu    memory         disk
#0   running   2014-11-10 05:07:33 PM   0.0%   686.4M of 1G   153.6M of 1G
#1   running   2014-11-10 05:07:36 PM   0.0%   677.2M of 1G   153.6M of 1G
```

Step 9: Test Your Deployed App

You've deployed an app to PAS!

Use the cf CLI or Apps Manager to review information and administer your app and your PAS account. For example, you can edit the `manifest.yml` to increase the number of app instances from 1 to 3, and redeploy the app with a new app name and host name.

See the Manage Your Application with the cf CLI section for more information. See also Using the Apps Manager.

Sample App Step
To test the sample app, do the following:

1. To export the test host, run `export HOST=SAMPLE-APP-URL`, substituting the URL for your app for `SAMPLE-APP-URL`.

2. To clear the database from any previous tests, run:
```
curl -v -X DELETE
$HOST/all
```
You should get a response of 200.

3. To request a match as “andrew”, run:
```
curl -v -H "Content-Type: application/json" -X PUT $HOST/match_requests/firstrequest -d '{"player":
"andrew"}'
```
You should again get a response of `200`.

4. To request a match as a different player, run:
```
curl -v -H "Content-Type: application/json" -X PUT $HOST/match_requests/secondrequest -d '{"player":
"navratilova"}'
```

5. To check the status of the first match request, run:
```
curl -v -X GET
$HOST/match_requests/firstrequest
```
The last line of the output shows the `match_id`.

6. Replace `MATCH_ID` with the `match_id` value from the previous step in the following command:
```
curl -v -H "Content-Type: application/json" -X POST $HOST/results -d ' { "match_id":"MATCH_ID", "winner":"andrew", "loser":"navratilova"
}'
```
You should receive a `201 Created` response.

Manage Your Application with the cf CLI

Run `cf help` to view a complete list of commands, grouped by task categories, and run `cf help COMMAND` for detailed information about a specific command. For more information about using the cf CLI, refer to the Cloud Foundry Command Line Interface (cf CLI) topics, especially the Getting Started with cf CLI topic.

Note: You cannot perform certain tasks in the CLI or Apps Manager because these are commands that only a PAS administrator can run. If you are not a PAS administrator, the following message displays for these types of commands: `error code: 10003, message: You are not authorized to perform the requested action` For more information about specific Admin commands you can perform with the Apps Manager, depending on your user role, refer to the Getting Started with the Apps Manager topic.

Troubleshooting

If your application fails to start, verify that the application starts in your local environment. Refer to the Troubleshooting Application Deployment and Health topic to learn more about troubleshooting.

App Deploy Fails

Even when the deploy fails, the app might exist on PAS. Run `cf apps` to review the apps in the currently targeted org and space. You might be able to correct the issue using the CLI or Apps Manager, or you might have to delete the app and redeploy.

App Requires a Unique URL

PAS requires that each app that you deploy have a unique URL. Otherwise, the new app URL collides with an existing app URL and PAS cannot successfully deploy the app. You can fix this issue by running `cf push` with either of the following flags to create a unique URL:

- `-n` to assign a different HOST name for the app.
- `--random-route` to create a URL that includes the app name and random words. Using this option might create a long URL, depending on the number of words that the app name includes.

Getting Started Deploying Ratpack Apps

Page last updated:

This guide is intended to walk you through deploying a Ratpack app to Pivotal Application Service. If you experience a problem following the steps below, check the Known Issues topic or refer to the Troubleshooting Application Deployment and Health topic.

Sample App Step
If you want to go through this tutorial using the sample app, run `git clone https://github.com/cloudfoundry-samples/pong_matcher_groovy.git` to clone the `pong_matcher_groovy` app from GitHub, and follow the instructions in the Sample App Step sections.

Note: Ensure that your Ratpack app runs locally before continuing with this procedure.

Deploy a Ratpack Application

This section describes how to deploy a Ratpack application to PAS.

Prerequisites

- A Ratpack app that runs locally on your workstation
- Intermediate to advanced Ratpack knowledge
- The Cloud Foundry Command Line Interface (cf CLI)
- JDK 1.7 or 1.8 for Java 7 or 8 configured on your workstation

Note: You can develop Ratpack applications in Java 7 or 8 or any JVM language. The Cloud Foundry Java buildpack uses JDK 1.8, but you can modify the buildpack and the manifest for your app to compile to JDK 1.7. Refer to Step 8: Configure the Deployment Manifest.

Step 1: Declare App Dependencies

Declare all the dependency tasks for your app in the build script of your chosen build tool. The table lists build script information for Gradle and Maven and provides documentation links for each build tool.

Build Tool	Build Script	Documentation
Gradle	`build.gradle`	Gradle User Guide
Maven	`pom.xml`	Apache Maven Project Documentation

Sample App Step
You can skip this step. The `build.gradle` file contains the dependencies for the `pong_matcher_groovy` sample app, as the example below shows.

```
dependencies {
  // SpringLoaded enables runtime hot reloading.
  // It is not part of the app runtime and is not shipped in the distribution.
  springloaded "org.springframework:springloaded:1.2.0.RELEASE"

  // Default SLF4J binding.  Note that this is a blocking implementation.
  // See here for a non blocking appender http://logging.apache.org/log4j/2.x/manual/async.html
  runtime 'org.slf4j:slf4j-simple:1.7.7'

  compile group: 'redis.clients', name: 'jedis', version: '2.5.2', transitive: true

  testCompile "org.spockframework:spock-core:0.7-groovy-2.0"
}
```

Step 2: Allocate Sufficient Memory

Use the `cf push -m` command to specify the amount of memory that should be allocated to the application. Memory allocated this way is done in preset amounts of `64M`, `128M`, `256M`, `512M`, `1G`, or `2G`. For example:

```
$ cf push -m 128M
```

When your app is running, you can use the `cf app APP-NAME` command to see memory utilization.

Sample App Step
You can skip this step. In the `manifest.yml` of the `pong_matcher_groovy` sample app, the `memory` sub-block of the `applications` block allocates 512 MB to the app.

Step 3: Provide a JDBC Driver

The Java buildpack does not bundle a JDBC driver with your application. If your application accesses a SQL RDBMS, you must do the following:

- Include the appropriate driver in your application.
- Create a dependency task for the driver in the build script for your build tool or IDE.

Sample App Step
You can skip this step. The `pong_matcher_groovy` sample app does not require a JDBC driver.

Step 4: (Optional) Configure a Procfile

Use a Procfile to declare required runtime processes for your web app and to specify your web server. For more information, see the Configuring a Production Server topic.

Sample App Step
You can skip this step. The `pong_matcher_groovy` app does not require a Procfile.

Step 5: Create and Bind a Service Instance for a Ratpack Application

This section describes using the CLI to configure a Redis managed service instance for an app. You can use either the CLI or Apps Manager to perform this task.

PAS supports two types of service instances:

- Managed services integrate with PAS through service brokers that offer services and plans and manage the service calls between PAS and a service provider.
- User-provided service instances enable you to connect your application to pre-provisioned external service instances.

For more information about creating and using service instances, refer to the Services Overview topic.

Create a Service Instance

Run `cf marketplace` to view managed and user-provided services and plans available to you.

The example shows two of the available managed database-as-a-service providers and their offered plans: `elephantsql` PostgreSQL as a Service and `rediscloud` Enterprise-Class Redis for Developers.

```
$ cf marketplace
Getting services from marketplace in org Cloud-Apps / space development as clouduser@example.com...
OK

service      plans                    description

elephantsql  turtle, panda, elephant      PostgreSQL as a Service

rediscloud   30mb, 100mb, 1gb, 10gb, 50gb  Enterprise-Class Redis for Developers
```

Run `cf create-service SERVICE PLAN SERVICE-INSTANCE` to create a service instance for your app. Choose a SERVICE and PLAN from the list, and provide a unique name for the SERVICE-INSTANCE.

Sample App Step
Run `cf create-service rediscloud 30mb baby-redis`. This creates a service instance named `baby-redis` that uses the `rediscloud` service and the `30mb` plan, as the example below shows.

```
$ cf create-service rediscloud 30mb baby-redis
Creating service baby-redis in org Cloud-Apps / space development as clouduser@example.com....
OK
```

Bind a Service Instance

When you bind an app to a service instance, PAS writes information about the service instance to the VCAP_SERVICES app environment variable. The app can use this information to integrate with the service instance.

Most services support bindable service instances. Refer to your service provider's documentation to confirm if they support this functionality.

You can bind a service to an application with the command `cf bind-service APPLICATION SERVICE_INSTANCE`.

Alternately, you can configure the deployment manifest file by adding a `services` sub-block to the `applications` block and specifying the service instance. For more information and an example on service binding using a manifest, see the Sample App step.

You can also bind a service using the Apps Manager.

Sample App Step
You can skip this step because the service instance is already bound. Open the `manifest.yml` file in a text editor to view the bound service instance information. Locate the file in the app root directory and search for the `services` sub-block in the `applications` block, as the example below shows.

```
---
applications:
...
  services:
    - baby-redis
```

Step 6: Configure the Deployment Manifest

You can specify deployment options in the `manifest.yml` that the `cf push` command uses when deploying your app.

Refer to the Deploying with Application Manifests topic for more information.

Sample App Step
You can skip this step. The `manifest.yml` file for the `pong_matcher_groovy` sample app does not require any additional configuration to deploy the app.

Step 7: Log in and Target the API Endpoint

Run `cf login -a API-ENDPOINT`, enter your login credentials, and select a space and org. The API endpoint is the URL of the Cloud Controller in your PAS instance.

Sample App Step
You must perform this step to run the sample app.

Step 8: Deploy the Application

Note: You must use the cf CLI to deploy apps.

From the root directory of your application, run `cf push APP-NAME -p PATH-TO-FILE.distZip` to deploy your application.

Note: You must deploy the `.distZip` artifact for a Ratpack app, and you must include the path to the `.distZip` file in the `cf push` command using the `-p` option if you do not declare the path in the `applications` block of the manifest file. For more information, refer to the Tips for Java Developers topic.

`cf push APP-NAME` creates a URL route to your application in the form HOST.DOMAIN, where HOST is your APP-NAME and DOMAIN is specified by your administrator. Your DOMAIN is `shared-domain.example.com`. For example: `cf push my-app` creates the URL `my-app.shared-domain.example.com`.

The URL for your app must be unique from other apps that PAS hosts or the push will fail. Use the following options to help create a unique URL:

- `-n` to assign a different HOST name for the app
- `--random-route` to create a URL that includes the app name and random words
- `cf help push` to view other options for this command

If you want to view log activity while the app deploys, launch a new terminal window and run `cf logs APP-NAME`.

Once your app deploys, browse to your app URL. Search for the `urls` field in the `App started` block in the output of the `cf push` command. Use the URL to access your app online.

Sample App Step
1. Change to the `app` directory, and run `./gradlew distZip` to build the app.
2. Run `cf push pong_matcher_groovy -n HOST-NAME` to push the app.

Example: `cf push pong_matcher_groovy -n groovy-ratpack-app`

Note: You do not have to include the `-p` flag when you deploy the sample app. The sample app manifest declares the path to the archive that `cf push` uses to upload the app files.

The example below shows the terminal output of deploying the `pong_matcher_groovy` app. `cf push` uses the instructions in the manifest file to create the app, create and bind the route, and upload the app. It then binds the app to the `baby-redis` service and follows the instructions in the manifest to start one instance of the app with 512 MB. After the app starts, the output displays the health and status of the app.

```
$ cf push pong_matcher_groovy -n groovy-ratpack-app
Using manifest file /Users/example/workspace/pong_matcher_groovy/app/manifest.yml

Creating app pong_matcher_groovy in org Cloud-Apps / space development as clouduser@example.com...
OK

Creating route groovy-ratpack-app.cfapps.io...
OK

Binding groovy-ratpack-app.cfapps.io to pong_matcher_groovy...
OK

Uploading pong_matcher_groovy...
Uploading app files from: /Users/example/workspace/pong_matcher_groovy/app/build/distributions/app.zip
Uploading 138.2K, 18 files
OK
Binding service baby-redis to app pong_matcher_groovy in org Cloud-Apps / space development as clouduser@example.com...
OK

Starting app pong_matcher_groovy in org Cloud-Apps / space development as clouduser@example.com...
OK
-----> Downloaded app package (12M)
Cloning into '/tmp/buildpacks/java-buildpack'...
-----> Java Buildpack Version: 9e096be | https://github.com/cloudfoundry/java-buildpack#9e096be
       Expanding Open Jdk JRE to .java-buildpack/open_jdk_jre (1.3s)
-----> Uploading droplet (49M)

0 of 1 instances running, 1 starting
1 of 1 instances running

App started

Showing health and status for app pong_matcher_groovy in org Cloud-Apps / space development as clouduser@example.com...
OK

requested state: started
instances: 1/1
usage: 512M x 1 instances
urls: groovy-ratpack-app.cfapps.io

     state     since                    cpu    memory          disk
#0   running   2014-10-28 04:48:58 PM   0.0%   193.5M of 512M   111.7M of 1G
```

Step 9: Test Your Deployed App

You've deployed an app to PAS!

Use the cf CLI or Apps Manager to review information and administer your app and your PAS account. For example, you can edit the `manifest.yml` to increase the number of app instances from 1 to 3, and redeploy the app with a new app name and host name.

See the Manage Your Application with the cf CLI section for more information. See also Using the Apps Manager.

Sample App Step

To test the sample app, do the following:

1. To export the test host, run `export HOST=SAMPLE-APP-URL`, substituting the URL for your app for `SAMPLE-APP-URL`.

2. To clear the database from any previous tests, run:
```
curl -v -X DELETE
$HOST/all
```
You should get a response of 200.

3. To request a match as "andrew", run:
```
curl -v -H "Content-Type: application/json" -X PUT $HOST/match_requests/firstrequest -d '{"player":
"andrew"}'
```
You should again get a response of `200`.

4. To request a match as a different player, run:
```
curl -v -H "Content-Type: application/json" -X PUT $HOST/match_requests/secondrequest -d '{"player":
"navratilova"}'
```

5. To check the status of the first match request, run:

```
curl -v -X GET
$HOST/match_requests/firstrequest
```

The last line of the output shows the `match_id`.

6. Replace `MATCH_ID` with the `match_id` value from the previous step in the following command:

```
curl -v -H "Content-Type: application/json" -X POST $HOST/results -d ' { "match_id":"MATCH_ID", "winner":"andrew", "loser":"navratilova"
}'
```

You should receive a `201 Created` response.

Manage Your Application with the cf CLI

Run `cf help` to view a complete list of commands, grouped by task categories, and run `cf help COMMAND` for detailed information about a specific command. For more information about using the cf CLI, refer to the Cloud Foundry Command Line Interface (cf CLI) topics, especially the Getting Started with cf CLI topic.

Note: You cannot perform certain tasks in the CLI or Apps Manager because these are commands that only a PAS administrator can run. If you are not a PAS administrator, the following message displays for these types of commands: `error code: 10003, message: You are not authorized to perform the requested action` For more information about specific Admin commands you can perform with the Apps Manager, depending on your user role, refer to the Getting Started with the Apps Manager topic.

Troubleshooting

If your application fails to start, verify that the application starts in your local environment. Refer to the Troubleshooting Application Deployment and Health topic to learn more about troubleshooting.

App Deploy Fails

Even when the deploy fails, the app might exist on PAS. Run `cf apps` to review the apps in the currently targeted org and space. You might be able to correct the issue using the CLI or Apps Manager, or you might have to delete the app and redeploy.

App Requires a Unique URL

PAS requires that each app that you deploy have a unique URL. Otherwise, the new app URL collides with an existing app URL and PAS cannot successfully deploy the app. You can fix this issue by running `cf push` with either of the following flags to create a unique URL:

- `-n` to assign a different HOST name for the app.
- `--random-route` to create a URL that includes the app name and random words. Using this option might create a long URL, depending on the number of words that the app name includes.

Getting Started Deploying Spring Apps

Page last updated:

This guide is intended to walk you through deploying a Spring app to Pivotal Application Service. You can choose whether to push a sample app, your own app, or both.

If you experience a problem following the steps below, see the Known Issues topic, or refer to the Troubleshooting Application Deployment and Health topic.

Sample App Step
If you want to go through this tutorial using the sample app, run `git clone https://github.com/cloudfoundry-samples/pong_matcher_spring` to clone the `pong_matcher_spring` app from GitHub, and follow the instructions in the Sample App Step sections.

Note: Ensure that your Spring app runs locally before continuing with this procedure.

Deploy a Spring Application

This section describes how to deploy your Spring application to PAS.

Prerequisites

- A Spring app that runs locally on your workstation
- Intermediate to advanced Spring knowledge
- The Cloud Foundry Command Line Interface (cf CLI)
- JDK 1.6, 1.7, or 1.8 for Java 6, 7, or 8 configured on your workstation

Note: The Cloud Foundry Java buildpack uses JDK 1.8, but you can modify the buildpack and the manifest for your app to compile to an earlier version. For more information, refer to the Custom Buildpacks topic.

Step 1: Declare App Dependencies

Be sure to declare all the dependency tasks for your app in the build script of your chosen build tool.

The Spring Getting Started Guides demonstrate features and functionality you can add to your app, such as consuming RESTful services or integrating data. These guides contain Gradle and Maven build script examples with dependencies. You can copy the code for the dependencies into your build script.

The table lists build script information for Gradle and Maven and provides documentation links for each build tool.

Build Tool	Build Script	Documentation
Gradle	`build.gradle`	Gradle User Guide
Maven	`pom.xml`	Apache Maven Project Documentation

Sample App Step
You can skip this step. The `pom.xml` file contains the dependencies for the `pong_matcher_spring` sample app, as the example below shows.

```
<dependencies>
    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
    </dependency>
    <dependency>
      <groupId>org.flywaydb</groupId>
      <artifactId>flyway-core</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
      <groupId>com.h2database</groupId>
      <artifactId>h2</artifactId>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>com.jayway.jsonpath</groupId>
      <artifactId>json-path</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>
```

Note: Make sure you are not building fully executable jars because application push may fail.

Step 2: Allocate Sufficient Memory

Use the `cf push -m` command to specify the amount of memory that should be allocated to the application. Memory allocated this way is done in preset amounts of `64M`, `128M`, `256M`, `512M`, `1G`, or `2G`. For example:

```
$ cf push -m 128M
```

When your app is running, you can use the `cf app APP-NAME` command to see memory utilization.

Sample App Step
You can skip this step. The Cloud Foundry Java buildpack uses settings declared in the sample app to allocate 1 GB of memory to the app.

Step 3: Provide a JDBC Driver

The Java buildpack does not bundle a JDBC driver with your application. If your application accesses a SQL RDBMS, you must do the following:

- Include the appropriate driver in your application.
- Create a dependency task for the driver in the build script for your build tool or IDE.

Sample App Step
You can skip this step. In the `pong_matcher_spring` sample app, the `src/main/resources/application.yml` file declares the JDBC driver, and the `pom.xml` file includes the JDBC driver as a dependency.

Step 4: Configure Service Connections for a Spring App

PAS provides extensive support for creating and binding a Spring application to services such as MySQL, PostgreSQL, MongoDB, Redis, and RabbitMQ. For more information about creating and binding a service connection for your app, refer to the Configure Service Connections for Spring topic.

Sample App Step: Create a Service Instance
Run `cf create-service cleardb spark mysql`. This creates a service instance named `mysql` that uses the `cleardb` service and the `spark` plan, as the example below shows.

```
$ cf create-service cleardb spark mysql
Creating service mysql in org Cloud-Apps / space development as a.user@example.com....
OK
```

Sample App Step: Bind a Service Instance
You can skip this step because the service instance is already bound. Open the `manifest.yml` file in a text editor to view the bound service instance information. Locate the file in the app root directory and search for the `services` sub-block in the `applications` block, as the example below shows.

```
---
applications:
...
  services:
    - mysql
```

Step 5: Configure the Deployment Manifest

You can specify deployment options in a manifest file `manifest.yml` that the `cf push` command uses when deploying your app.

Refer to the Deploying with Application Manifests topic for more information.

Sample App Step
You can skip this step. The `manifest.yml` file for the `pong_matcher_spring` sample app does not require any additional configuration to deploy the app.

Step 6: Log in and Target the API Endpoint

Run `cf login -a API-ENDPOINT`, enter your login credentials, and select a space and org. The API endpoint is the URL of the Cloud Controller in your PAS instance.

Sample App Step
You must do this step to run the sample app.

Step 7: Deploy Your Application

Note: You must use the cf CLI to deploy apps.

From the root directory of your application, run `cf push APP-NAME -p PATH-TO-FILE.war` to deploy your application.

Note: Most Spring apps include an artifact, such as a `.jar`, `.war`, or `.zip` file. You must include the path to this file in the `cf push` command using the `-p` option if you do not declare the path in the `applications` block of the manifest file. The example shows how to specify a path to the `.war` file for a Spring app. Refer to the Tips for Java Developers topic for CLI examples for specific build tools, frameworks, and languages that create an app with an artifact.

`cf push APP-NAME` creates a URL route to your application in the form HOST.DOMAIN, where HOST is your APP-NAME and DOMAIN is specified by your administrator. Your DOMAIN is `shared-domain.example.com`. For example: `cf push my-app` creates the URL `my-app.shared-domain.example.com`.

The URL for your app must be unique from other apps that PAS hosts or the push will fail. Use the following options to help create a unique URL:

- `-n` to assign a different HOST name for the app
- `--random-route` to create a URL that includes the app name and random words
- `cf help push` to view other options for this command

If you want to view log activity while the app deploys, launch a new terminal window and run `cf logs APP-NAME`.

Once your app deploys, browse to your app URL. Search for the `urls` field in the `App started` block in the output of the `cf push` command. Use the URL to access your app online.

Sample App Step

1. Run `brew install maven`.
2. Change to the `app` directory, and run `mvn package` to build the app.
3. Run `cf push pong_matcher_spring -n HOSTNAME` to push the app.

Example: `cf push pong_matcher_spring -n my-spring-app`

Note: You do not have to include the `-p` flag when you deploy the sample app. The sample app manifest declares the path to the archive that `cf push` uses to upload the app files.

The example below shows the terminal output of deploying the `pong_matcher_spring` app. `cf push` uses the instructions in the manifest file to create the app, create and bind the route, and upload the app. It then binds the app to the `mysql` service and starts one instance of the app with 1 GB of memory. After the app starts, the output displays the health and status of the app.

```
$ cf push pong_matcher_spring -n spring1119
Using manifest file /Users/example/workspace/pong_matcher_spring/manifest.yml

Creating app pong_matcher_spring in org Cloud-Apps / space development as a.user@example.com...
OK

Creating route spring1119.cfapps.io...
OK

Binding spring1119.cfapps.io to pong_matcher_spring...
OK

Uploading pong_matcher_spring...
Uploading app files from: /Users/example/workspace/pong_matcher_spring/target/pong-matcher-spring-1.0.0.BUILD-SNAPSHOT.jar
Uploading 797.5K, 116 files
OK
Binding service mysql to app pong_matcher_spring in org Cloud-Apps / space development as a.user@example.com...
OK

Starting app pong_matcher_spring in org Cloud-Apps / space development as a.user@example.com...
OK
-----> Downloaded app package (25M)
-----> Downloading Open Jdk JRE 1.8.0_25 from https://download.run.pivotal.io/openjdk/lucid/x86_64/openjdk-1.8.0_25.tar.gz (1.2s)
       Expanding Open Jdk JRE to .java-buildpack/open_jdk_jre (1.1s)
-----> Downloading Spring Auto Reconfiguration 1.5.0_RELEASE from https://download.run.pivotal.io/auto-reconfiguration/auto-reconfiguration-1.5.0_RELEASE.jar (0.1s)

-----> Uploading droplet (63M)

0 of 1 instances running, 1 starting
1 of 1 instances running

App started

Showing health and status for app pong_matcher_spring in org Cloud-Apps / space development as a.user@example.com...
OK

requested state: started
instances: 1/1
usage: 1G x 1 instances
urls: spring1119.cfapps.io

     state     since                    cpu    memory        disk
#0   running   2014-11-19 12:29:27 PM   0.0%   553.6M of 1G   127.4M of 1G
```

Step 8: Test Your Deployed App

You've deployed an app to PAS!

Use the cf CLI or Apps Manager to review information and administer your app and your PAS account. For example, you can edit the `manifest.yml` to increase the number of app instances from 1 to 3, and redeploy the app with a new app name and host name.

See the Manage Your Application with the cf CLI section for more information. See also Using the Apps Manager.

Sample App Step

To test the sample app, do the following:

1. To export the test host, run `export HOST=SAMPLE-APP-URL`, substituting the URL for your app for `SAMPLE-APP-URL`.

2. To clear the database from any previous tests, run:

```
curl -v -X DELETE
$HOST/all
```

You should get a response of 200.

3. To request a match as "andrew", run:

```
curl -v -H "Content-Type: application/json" -X PUT $HOST/match_requests/firstrequest -d '{"player":
"andrew"}'
```

You should again get a response of `200`.

4. To request a match as a different player, run:

```
curl -v -H "Content-Type: application/json" -X PUT $HOST/match_requests/secondrequest -d '{"player":
"navratilova"}'
```

5. To check the status of the first match request, run:
```
curl -v -X GET
$HOST/match_requests/firstrequest
```
The last line of the output shows the `match_id`.

6. Replace `MATCH_ID` with the `match_id` value from the previous step in the following command:
```
curl -v -H "Content-Type: application/json" -X POST $HOST/results -d ' { "match_id":"MATCH_ID", "winner":"andrew", "loser":"navratilova"
}'
```
You should receive a `201 Created` response.

Alternative Method for Pushing Apps

You can also integrate the Cloud Foundry Eclipse Plugin for STS.

PAS provides an Eclipse plugin extension that enables you to deploy and manage Spring applications on a Cloud Foundry instance from the Spring Tool Suite (STS), version 3.0.0 and later. For more information, refer to the Cloud Foundry Eclipse Plugin topic. You must follow the instructions in the Install to STS from the IDE Extensions Tab and Create a Cloud Foundry Server sections before you can deploy and manage your apps with the plugin.

Manage Your Application with the cf CLI

Run `cf help` to view a complete list of commands, grouped by task categories, and run `cf help COMMAND` for detailed information about a specific command. For more information about using the cf CLI, refer to the Cloud Foundry Command Line Interface (cf CLI) topics, especially the Getting Started with cf CLI topic.

Note: You cannot perform certain tasks in the CLI or Apps Manager because these are commands that only a PAS administrator can run. If you are not a PAS administrator, the following message displays for these types of commands: `error code: 10003, message: You are not authorized to perform the requested action` For more information about specific Admin commands you can perform with the Apps Manager, depending on your user role, refer to the Getting Started with the Apps Manager topic.

Troubleshooting

If your application fails to start, verify that the application starts in your local environment. Refer to the Troubleshooting Application Deployment and Health topic to learn more about troubleshooting.

App Deploy Fails

Even when the deploy fails, the app might exist on PAS. Run `cf apps` to review the apps in the currently targeted org and space. You might be able to correct the issue using the CLI or Apps Manager, or you might have to delete the app and redeploy.

App Requires a Content-Type

If you specify a `Content-Encoding` header of `gzip` but do not specify a `Content-Type` within your application, PAS might send a `Content-Type` of `application/x-gzip` to the browser. This scenario might cause the deploy to fail if it conflicts with the actual encoded content of your app. To avoid this issue, be sure to explicitly set `Content-Type` within your app.

App Requires a Unique URL

PAS requires that each app that you deploy have a unique URL. Otherwise, the new app URL collides with an existing app URL and PAS cannot successfully deploy the app. You can fix this issue by running `cf push` with either of the following flags to create a unique URL:

- `-n` to assign a different HOST name for the app.

- `--random-route` to create a URL that includes the app name and random words. Using this option might create a long URL, depending on the number of words that the app name includes.

Configuring Service Connections

Page last updated:

This topic provides links to additional information about configuring service connections for Java apps. See the specific documentation for your app framework:

- Grails
- Play
- Spring

Configuring Service Connections for Grails

Page last updated:

Cloud Foundry provides extensive support for connecting a Grails application to services such as MySQL, Postgres, MongoDB, Redis, and RabbitMQ. In many cases, a Grails application running on Cloud Foundry can automatically detect and configure connections to services. For more advanced cases, you can control service connection parameters yourself.

Auto-Configuration

Grails provides plugins for accessing SQL (using Hibernate), MongoDB, and Redis services. If you install any of these plugins and configure them in your `Config.groovy` or `DataSource.groovy` file, Cloud Foundry reconfigures the plugin with connection information when your app starts.

If you use all three types of services, your configuration might look like this:

```
environments {
  production {
   dataSource {
     url = 'jdbc:mysql://localhost/db?useUnicode=true&characterEncoding=utf8'
     dialect = org.hibernate.dialect.MySQLInnoDBDialect
     driverClassName = 'com.mysql.jdbc.Driver'
     username = 'user'
     password = "password"
   }
   grails {
     mongo {
       host = 'localhost'
       port = 27107
       databaseName = "foo"
       username = 'user'
       password = 'password'
     }
     redis {
       host = 'localhost'
       port = 6379
       password = 'password'
       timeout = 2000
     }
   }
  }
}
```

The `url`, `host`, `port`, `databaseName`, `username`, and `password` fields in this configuration will be overriden by the Cloud Foundry auto-reconfiguration if it detects that the application is running in a Cloud Foundry environment. If you want to test the application locally against your own services, you can put real values in these fields. If the application will only be run against Cloud Foundry services, you can put placeholder values as shown here, but the fields must exist in the configuration.

Manual Configuration

If you do not want to use auto-configuration, you can configure the Cloud Foundry service connections manually.

Follow the steps below to manually configure a service connection.

1. Add the `spring-cloud` library to the `dependencies` section of your `BuildConfig.groovy` file.

   ```
   repositories {
     grailsHome()
     mavenCentral()
     grailsCentral()
     mavenRepo "http://repo.spring.io/milestone"
   }

   dependencies {
     compile "org.springframework.cloud:spring-cloud-cloudfoundry-connector:1.0.0.RELEASE"
     compile "org.springframework.cloud:spring-cloud-spring-service-connector:1.0.0.RELEASE"
   }
   ```

   Adding the `spring-cloud` library allows you to disable auto-configuration and use the `spring-cloud` API in your `DataSource.groovy` file to set the connection parameters.

2. Add the following to your `grails-app/conf/spring/resources.groovy` file to disable auto-configuration:

```
beans = {
    cloudFactory(org.springframework.cloud.CloudFactory)
}
```

3. Add the following `imports` to your `DataSource.groovy` file to allow `spring-cloud` API commands:

```
import org.springframework.cloud.CloudFactory
import org.springframework.cloud.CloudException
```

4. Add the following code to your `DataSource.groovy` file to enable Cloud Foundry's `getCloud` method to function locally or in other environments outside of a cloud.

```
def cloud = null
try {
    cloud = new CloudFactory().cloud
} catch (CloudException) {}
```

5. Use code like the following to access the cloud object:

```
def dbInfo = cloud?.getServiceInfo('myapp-mysql')
url = dbInfo?.jdbcUrl
username = dbInfo?.userName
password = dbInfo?.password
```

`myapp-mysql` is the name of the service as it appears in the `name` column of the output from `cf services`. For example, `mysql` or `rabbitmq`.

The example `DataSource.groovy` file below contains the following:

- The `imports` that allow `spring-cloud` API commands
- The code that enables the `getCloud` method to function locally or in other environments outside of a cloud
- Code to access the cloud object for SQL, MongoDB, and Redis services

```
import org.springframework.cloud.CloudFactory
import org.springframework.cloud.CloudException

def cloud = null
try {
    cloud = new CloudFactory().cloud
} catch (CloudException) {}

dataSource {
    pooled = true
    dbCreate = 'update'
    driverClassName = 'com.mysql.jdbc.Driver'
}

environments {
    production {
        dataSource {
            def dbInfo = cloud?.getServiceInfo('myapp-mysql')
            url = dbInfo?.jdbcUrl
            username = dbInfo?.userName
            password = dbInfo?.password
        }
        grails {
            mongo {
                def mongoInfo = cloud?.getServiceInfo('myapp-mongodb')
                host = mongoInfo?.host
                port = mongoInfo?.port
                databaseName = mongoInfo?.database
                username = mongoInfo?.userName
                password = mongoInfo?.password
            }
            redis {
                def redisInfo = cloud?.getServiceInfo('myapp-redis')
                host = redisInfo?.host
                port = redisInfo?.port
                password = redisInfo?.password
            }
        }
    }
    development {
        dataSource {
            url = 'jdbc:mysql://localhost:5432/myapp'
            username = 'sa'
            password = ''
        }
        grails {
            mongo {
                host = 'localhost'
                port = 27107
                databaseName = 'foo'
                username = 'user'
                password = 'password'
            }
            redis {
                host = 'localhost'
                port = 6379
                password = 'password'
            }
        }
    }
}
```

Configuring Service Connections for Play Framework

Page last updated:

Cloud Foundry supports running Play Framework applications and the Play JPA plugin for auto-configuration for Play versions up to and including v2.4.x.

Cloud Foundry provides support for connecting a Play Framework application to services such as MySQL and Postgres. In many cases, a Play Framework application running on Cloud Foundry can automatically detect and configure connections to services.

Auto-Configuration

By default, Cloud Foundry detects service connections in a Play Framework application and configures them to use the credentials provided in the Cloud Foundry environment. Note that auto-configuration happens only if there is a single service of either of the supported types—MySQL or Postgres.

Configuring Service Connections for Spring

Page last updated:

Cloud Foundry provides extensive support for connecting a Spring application to services such as MySQL, PostgreSQL, MongoDB, Redis, and RabbitMQ. In many cases, Cloud Foundry can automatically configure a Spring application without any code changes. For more advanced cases, you can control service connection parameters yourself.

Auto-Reconfiguration

If your Spring application requires services such as a relational database or messaging system, you might be able to deploy your application to Cloud Foundry without changing any code. In this case, Cloud Foundry automatically re-configures the relevant bean definitions to bind them to cloud services.

For information about connecting to services from a Spring application, see Spring Cloud Spring Service Connector.

Cloud Foundry auto-reconfigures applications only if the following items are true for your application:

- Only one service instance of a given service type is bound to the application. In this context, different relational databases services are considered the same service type. For example, if both a MySQL and a PostgreSQL service are bound to the application, auto-reconfiguration does not occur.
- Only one bean of a matching type is in the Spring application context. For example, you can have only one bean of type `javax.sql.DataSource`.

With auto-reconfiguration, Cloud Foundry creates the `DataSource` or connection factory bean itself, using its own values for properties such as host, port, username and so on. For example, if you have a single `javax.sql.DataSource` bean in your application context that Cloud Foundry auto-reconfigures and binds to its own database service, Cloud Foundry does not use the username, password and driver URL you originally specified. Instead, it uses its own internal values. This is transparent to the application, which really only cares about having a ` DataSource where it can write data but does not really care what the specific properties are that created the database. Also, if you have customized the configuration of a service, such as the pool size or connection properties, Cloud Foundry auto-reconfiguration ignores the customizations.

For more information about auto-reconfiguration of specific services types, see the Service-Specific Details section.

Manual Configuration

Use manual configuration if you have multiple services of a given type or you want to have more control over the configuration than auto-reconfiguration provides.

To use manual configuration, include the `spring-cloud` library in your list of application dependencies. Update your application Maven `pom.xml` or Gradle `build.gradle` file to include dependencies on the `org.springframework.cloud:spring-cloud-spring-service-connector` and `org.springframework.cloud:spring-cloud-cloudfoundry-connector` artifacts.

For example, if you use Maven to build your application, the following `pom.xml` snippet shows how to add this dependency.

```
<dependencies>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-spring-service-connector</artifactId>
    <version>1.2.3.RELEASE</version>
  </dependency>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-cloudfoundry-connector</artifactId>
    <version>1.2.3.RELEASE</version>
  </dependency>
</dependencies>
```

You also need to update your application build file to include the Spring Framework Milestone repository. The following `pom.xml` snippet shows how to do this for Maven:

```
<repositories>
   <repository>
      <id>repository.springsource.milestone</id>
      <name>SpringSource Milestone Repository</name>
      <url>http://repo.springsource.org/milestone</url>
   </repository>
    ...
</repositories>
```

Java Configuration

Typical use of Java config involves extending the `AbstractCloudConfig` class and adding methods with the `@Bean` annotation to create beans for services. Apps migrating from auto-reconfiguration might first try Scanning for Services until they need more explicit control. Java config also offers a way to expose application and service properties. Use this for debugging or to create service connectors using a lower-level access.

Create a Service Bean

In the following example, the configuration creates a `DataSource` bean connecting to the only relational database service bound to the app. It also creates a `MongoDbFactory` bean, again, connecting to the only MongoDB service bound to the app. Check Javadoc for `AbstractCloudConfig` for ways to connect to other services.

```
class CloudConfig extends AbstractCloudConfig {
      @Bean
      public DataSource inventoryDataSource() {
         return connectionFactory().dataSource();
      }
      ... more beans to obtain service connectors
}
```

The bean names match the method names unless you specify an explicit value to the annotation such as `@Bean("inventory-service")`, following Spring's Java configuration standards.

If you have more than one service of a type bound to the app or want to have an explicit control over the services to which a bean is bound, you can pass the service names to methods such as `dataSource()` and `mongoDbFactory()` as follows:

```
class CloudConfig extends AbstractCloudConfig {

   @Bean
   public DataSource inventoryDataSource() {
      return connectionFactory().dataSource("inventory-db-service");
   }

   @Bean
   public MongoDbFactory documentMongoDbFactory() {
      return connectionFactory().mongoDbFactory("document-service");
   }

   ... more beans to obtain service connectors

}
```

Method such as `dataSource()` come in an additional overloaded variant that offer specifying configuration options such as the pooling parameters. See Javadoc for more details.

Connect to Generic Services

Java config supports access to generic services through the `service()` method. Generic services do not have a directly mapped method. This is typical for a newly introduced service or when connecting to a private service in private PaaS. The generic `service()` method follows the same pattern as the `dataSource()`, except it allows supplying the connector type as an additional parameters.

Scan for Services

You can scan for each bound service using the `@ServiceScan` annotation as shown below. This is conceptually similar to the @ComponentScan annotation in Spring:

```
@Configuration
@ServiceScan
class CloudConfig {

}
```

Here, one bean of the appropriate type (`DataSource` for a relational database service, for example) is created. Each created bean will have the `id` matching the corresponding service name. You can then inject such beans using auto-wiring:

```
@Autowired DataSource inventoryDb;
```

If the app is bound to more than one services of a type, you can use the `@Qualifier` annotation supplying it the name of the service as in the following code:

```
@Autowired @Qualifier("inventory-db") DataSource inventoryDb;
@Autowired @Qualifier("shipping-db") DataSource shippingDb;
```

Access Service Properties

You can expose raw properties for all services and the app through a bean as follows:

```
class CloudPropertiesConfig extends AbstractCloudConfig {

  @Bean
  public Properties cloudProperties() {
      return properties();
  }
}
```

Cloud Profile

Spring Framework versions 3.1 and above support bean definition profiles as a way to conditionalize the application configuration so that only specific bean definitions are activated when a certain condition is true. Setting up such profiles makes your application portable to many different environments so that you do not have to manually change the configuration when you deploy it to, for example, your local environment and then to Cloud Foundry.

See the Spring Framework documentation for additional information about using Spring bean definition profiles.

When you deploy a Spring application to Cloud Foundry, Cloud Foundry automatically enables the `cloud` profile.

Note: Cloud Foundry auto-reconfiguration requires the Spring application to be version 3.1 or later and include the Spring context JAR. If you are using an earlier version, update your framework or use a custom buildpack.

Profiles in Java Configuration

The `@Profile` annotation can be placed on `@Configuration` classes in a Spring application to set conditions under which configuration classes are invoked. By using the `default` and `cloud` profiles to determine whether the application is running on Cloud Foundry or not, your Java configuration can support both local and cloud deployments using Java configuration classes like these:

```
public class Configuration {
  @Configuration
  @Profile("cloud")
  static class CloudConfiguration {

    @Bean
    public DataSource dataSource() {
      CloudFactory cloudFactory = new CloudFactory();
      Cloud cloud = cloudFactory.getCloud();
      String serviceID = cloud.getServiceID();
      return cloud.getServiceConnector(serviceID, DataSource.class, null);
    }
  }

  @Configuration
  @Profile("default")
  static class LocalConfiguration {

  @Bean
    public DataSource dataSource() {
      BasicDataSource dataSource = new BasicDataSource();
      dataSource.setUrl("jdbc:postgresql://localhost/db");
      dataSource.setDriverClassName("org.postgresql.Driver");
      dataSource.setUsername("postgres");
      dataSource.setPassword("postgres");
      return dataSource;
    }
  }
}
```

Property Placeholders

Cloud Foundry exposes a number of application and service properties directly into its deployed applications. The properties exposed by Cloud Foundry include basic information about the application, such as its name and the cloud provider, and detailed connection information for all services currently bound to the application.

Service properties generally take one of the following forms:

```
cloud.services.{service-name}.connection.{property}
cloud.services.{service-name}.{property}
```

In this form, `{service-name}` refers to the name you gave the service when you bound it to your application at deploy time, and `{property}` is a field in the credentials section of the `VCAP_SERVICES` environment variable.

For example, assume that you created a Postgres service called `my-postgres` and then bound it to your application. Assume also that this service exposes credentials in `VCAP_SERVICES` as discrete fields. Cloud Foundry exposes the following properties about this service:

```
cloud.services.my-postgres.connection.hostname
cloud.services.my-postgres.connection.name
cloud.services.my-postgres.connection.password
cloud.services.my-postgres.connection.port
cloud.services.my-postgres.connection.username
cloud.services.my-postgres.plan
cloud.services.my-postgres.type
```

If the service exposed the credentials as a single `uri` field, then the following properties would be set up:

```
cloud.services.my-postgres.connection.uri
cloud.services.my-postgres.plan
cloud.services.my-postgres.type
```

For convenience, if you have bound just one service of a given type to your application, Cloud Foundry creates an alias based on the service type instead of the service name. For example, if only one MySQL service is bound to an application, the properties takes the form `cloud.services.mysql.connection.{property}`. Cloud Foundry uses the following aliases in this case:

- `mysql`
- `postgresql`
- `mongodb`

- `redis`
- `rabbitmq`

A Spring application can take advantage of these Cloud Foundry properties using the property placeholder mechanism. For example, assume that you have bound a MySQL service called `spring-mysql` to your application. Your application requires a `c3p0` connection pool instead of the connection pool provided by Cloud Foundry, but you want to use the same connection properties defined by Cloud Foundry for the MySQL service - in particular the username, password and JDBC URL.

The following table lists all the application properties that Cloud Foundry exposes to deployed applications.

Property	Description
`cloud.application.name`	The name provided when the application was pushed to Cloud Foundry.
`cloud.provider.url`	The URL of the cloud hosting the application, such as `cloudfoundry.com`.

The service properties that are exposed for each type of service are listed in the Service-Specific Details section.

Service-Specific Details

The following sections describe Spring auto-reconfiguration and manual configuration for the services supported by Cloud Foundry.

MySQL and Postgres

Auto-Reconfiguration

Auto-reconfiguration occurs if Cloud Foundry detects a `javax.sql.DataSource` bean in the Spring application context. The following snippet of a Spring application context file shows an example of defining this type of bean which Cloud Foundry will detect and potentially auto-reconfigure:

```
<bean class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close" id="dataSource">
  <property name="driverClassName" value="org.h2.Driver" />
  <property name="url" value="jdbc:h2:mem:" />
  <property name="username" value="sa" />
  <property name="password" value="" />
</bean>
```

The relational database that Cloud Foundry actually uses depends on the service instance you explicitly bind to your application when you deploy it: MySQL or Postgres. Cloud Foundry creates either a commons DBCP or Tomcat datasource depending on which datasource implementation it finds on the classpath.

Cloud Foundry internally generates values for the following properties: `driverClassName`, `url`, `username`, `password`, `validationQuery`.

Manual Configuration in Java

To configure a database service in Java configuration, create a `@Configuration` class with a `@Bean` method to return a `javax.sql.DataSource` bean. The bean can be created by helper classes in the `spring-cloud` library, as shown here:

```
@Configuration
public class DataSourceConfig {
  @Bean
  public DataSource dataSource() {
    CloudFactory cloudFactory = new CloudFactory();
    Cloud cloud = cloudFactory.getCloud();
    String serviceID = cloud.getServiceID();
    return cloud.getServiceConnector(serviceID, DataSource.class, null);
  }
}
```

MongoDB

Auto-Reconfiguration

You must use Spring Data MongoDB 1.0 M4 or later for auto-reconfiguration to work.

Auto-reconfiguration occurs if Cloud Foundry detects an `org.springframework.data.document.mongodb.MongoDbFactory` bean in the Spring application context. The following snippet of a Spring XML application context file shows an example of defining this type of bean which Cloud Foundry will detect and potentially auto-reconfigure:

```
<mongo:db-factory
  id="mongoDbFactory"
  dbname="pwdtest"
  host="127.0.0.1"
  port="1234"
  username="test_user"
  password="test_pass" />
```

Cloud Foundry creates a `SimpleMongoDbFactory` with its own values for the following properties: `host` , `port` , `username` , `password` , `dbname` .

Manual Configuration in Java

To configure a MongoDB service in Java configuration, create a `@Configuration` class with a `@Bean` method to return an `org.springframework.data.mongodb.MongoDbFactory` bean from Spring Data MongoDB. The bean can be created by helper classes in the `spring-cloud` library, as shown here:

```
@Configuration
public class MongoConfig {

  @Bean
  public MongoDbFactory mongoDbFactory() {
    CloudFactory cloudFactory = new CloudFactory();
    Cloud cloud = cloudFactory.getCloud();
    MongoServiceInfo serviceInfo = (MongoServiceInfo) cloud.getServiceInfo("my-mongodb");
    String serviceID = serviceInfo.getID();
    return cloud.getServiceConnector(serviceID, DataSource.class, null);
  }

  @Bean
  public MongoTemplate mongoTemplate() {
    return new MongoTemplate(mongoDbFactory());
  }
}
```

Redis

Auto-Configuration

You must be using Spring Data Redis 1.0 M4 or later for auto-configuration to work.

Auto-configuration occurs if Cloud Foundry detects a `org.springframework.data.redis.connection.RedisConnectionFactory` bean in the Spring application context. The following snippet of a Spring XML application context file shows an example of defining this type of bean which Cloud Foundry will detect and potentially auto-configure:

```
<bean id="redis"
  class="org.springframework.data.redis.connection.jedis.JedisConnectionFactory"
  p:hostName="localhost" p:port="6379" />
```

Cloud Foundry creates a `JedisConnectionFactory` with its own values for the following properties: `host` , `port` , `password` . This means that you must package the Jedis JAR in your application. Cloud Foundry does not currently support the JRedis and RJC implementations.

Manual Configuration in Java

To configure a Redis service in Java configuration, create a `@Configuration` class with a `@Bean` method to return an `org.springframework.data.redis.connection.RedisConnectionFactory` bean from Spring Data Redis. The bean can be created by helper classes in the `spring-cloud` library, as shown here:

```
@Configuration
public class RedisConfig {

    @Bean
    public RedisConnectionFactory redisConnectionFactory() {
        CloudFactory cloudFactory = new CloudFactory();
        Cloud cloud = cloudFactory.getCloud();
        RedisServiceInfo serviceInfo = (RedisServiceInfo) cloud.getServiceInfo("my-redis");
        String serviceID = serviceInfo.getID();
        return cloud.getServiceConnector(serviceID, RedisConnectionFactory.class, null);
    }

    @Bean
    public RedisTemplate redisTemplate() {
        return new StringRedisTemplate(redisConnectionFactory());
    }

}
```

RabbitMQ

Auto-Configuration

You must be using Spring AMQP 1.0 or later for auto-configuration to work. Spring AMQP provides publishing, multi-threaded consumer generation, and message conversion. It also facilitates management of AMQP resources while promoting dependency injection and declarative configuration.

Auto-configuration occurs if Cloud Foundry detects an `org.springframework.amqp.rabbit.connection.ConnectionFactory` bean in the Spring application context. The following snippet of a Spring application context file shows an example of defining this type of bean which Cloud Foundry will detect and potentially auto-configure:

```
<rabbit:connection-factory
    id="rabbitConnectionFactory"
    host="localhost"
    password="testpwd"
    port="1238"
    username="testuser"
    virtual-host="virthost" />
```

Cloud Foundry creates an `org.springframework.amqp.rabbit.connection.CachingConnectionFactory` with its own values for the following properties: `host`, `virtual-host`, `port`, `username`, `password`.

Manual Configuration in Java

To configure a RabbitMQ service in Java configuration, create a `@Configuration` class with a `@Bean` method to return an `org.springframework.amqp.rabbit.connection.ConnectionFactory` bean from the Spring AMQP library. The bean can be created by helper classes in the `spring-cloud` library, as shown here:

```
@Configuration
public class RabbitConfig {
    @Bean
    public ConnectionFactory rabbitConnectionFactory() {
        CloudFactory cloudFactory = new CloudFactory();
        Cloud cloud = cloudFactory.getCloud();
        AmqpServiceInfo serviceInfo = (AmqpServiceInfo) cloud.getServiceInfo("my-rabbit");
    String serviceID = serviceInfo.getID();
    return cloud.getServiceConnector(serviceID, ConnectionFactory.class, null);
    }

    @Bean
    public RabbitTemplate rabbitTemplate(ConnectionFactory connectionFactory) {
        return new RabbitTemplate(connectionFactory);
    }
}
```

Cloud Foundry Eclipse Plugin

Page last updated:

The Cloud Foundry Eclipse Plugin is an extension that enables Cloud Foundry users to deploy and manage Java and Spring applications on a Cloud Foundry instance from Eclipse or Spring Tool Suite (STS).

The plugin supports Eclipse v3.8 and v4.3 (a Java EE version is recommended), and STS 3.0.0 and later.

This page has instructions for installing and using v1.7.2 of the plugin.

You can use the plugin to perform the following actions:

- Deploy applications from an Eclipse or STS workspace to a running Cloud Foundry instance. The Cloud Foundry Eclipse plugin supports the following application types:
  - Spring Boot
  - Spring
  - Java Web
  - Java standalone
  - Grails
- Create, bind, and unbind services.
- View and manage deployed applications and services.
- Start and stop applications.

v1.7.2 of this plugin provides the following updates and changes:

- Cloud Foundry Eclipse is now enabled for NLS and Internationalization.
- A "New Service Binding" wizard that allows service instances to be bound to applications. This wizard serves as an alternative to binding through the existing drag-and-drop feature.
- Improvements in Loggregator streaming to the console.
- Improvements in deploying Spring Boot and Getting Started projects with templates.

Install Cloud Foundry Eclipse Plugin

If you have a previous version of the Cloud Foundry Eclipse Plugin installed, uninstall it before installing the new version. To uninstall the plugin:

1. Choose **About Eclipse** (or **About Spring Tool Suite**) from the Eclipse (or **Spring Tool Suite**) menu and click **Installation Details**.
2. In **Installation Details**, select the previous version of the plugin and click **Uninstall**.

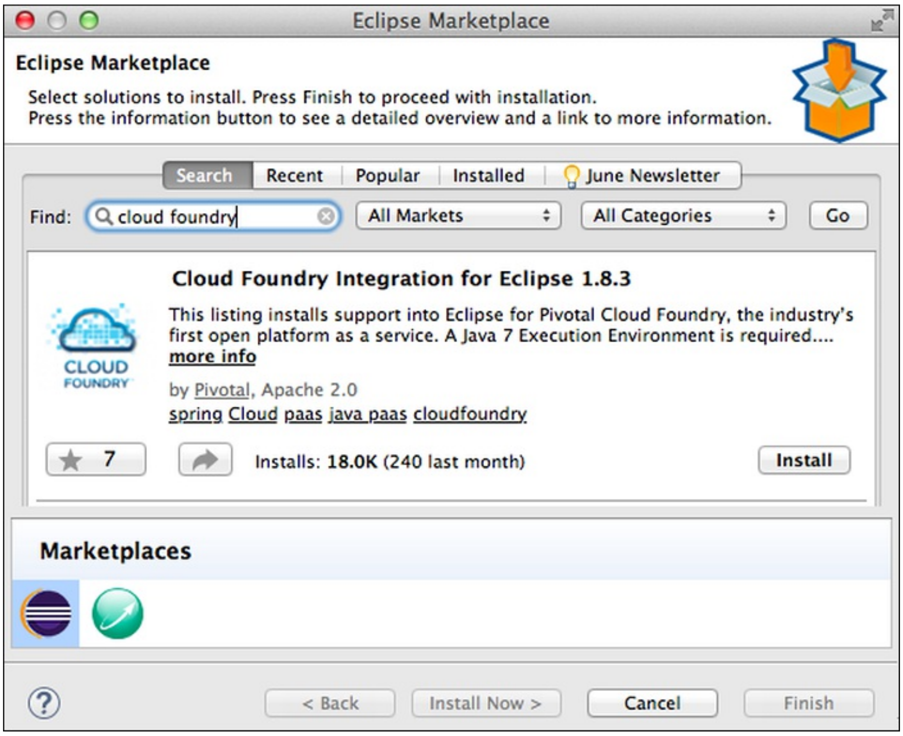
Follow the installation instructions appropriate for your environment:

- Install to Eclipse from Marketplace
- Install to STS from IDE Extensions Tab
- Install from a Local Repository

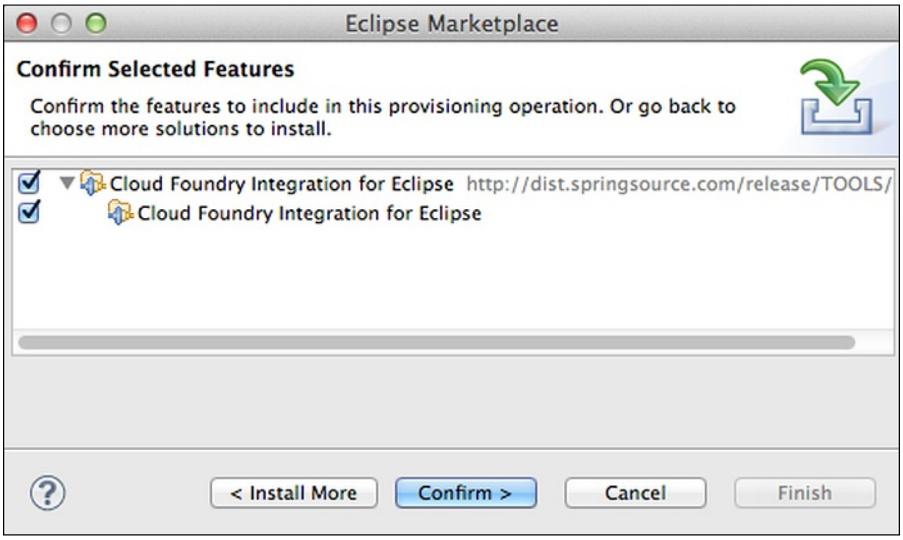
Install to Eclipse from Marketplace

Follow the instructions below to install the Cloud Foundry Eclipse Plugin to Eclipse from the Eclipse Marketplace.

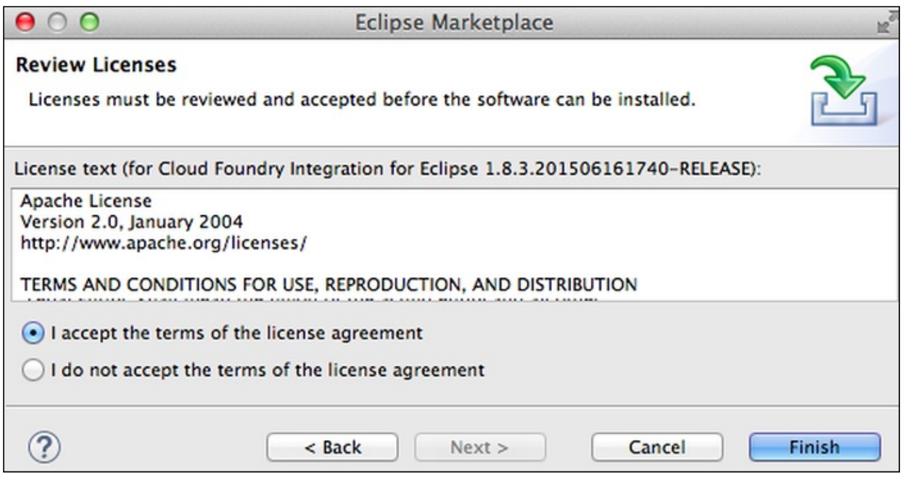
1. Start Eclipse.
2. From the Eclipse **Help** menu, select **Eclipse Marketplace**.
3. In the Eclipse Marketplace window, enter "Cloud Foundry" in the **Find** field. Click **Go**.
4. In the search results, next to the listing for Cloud Foundry Integration, click **Install**.



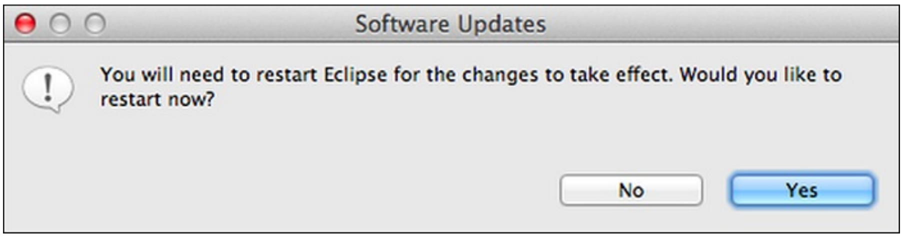
5. In the **Confirm Selected Features** window, click **Confirm**.



6. The **Review Licenses** window appears. Select “I accept the terms of the license agreement” and click **Finish**.



7. The **Software Updates** window appears. Click **Yes** to restart Eclipse.

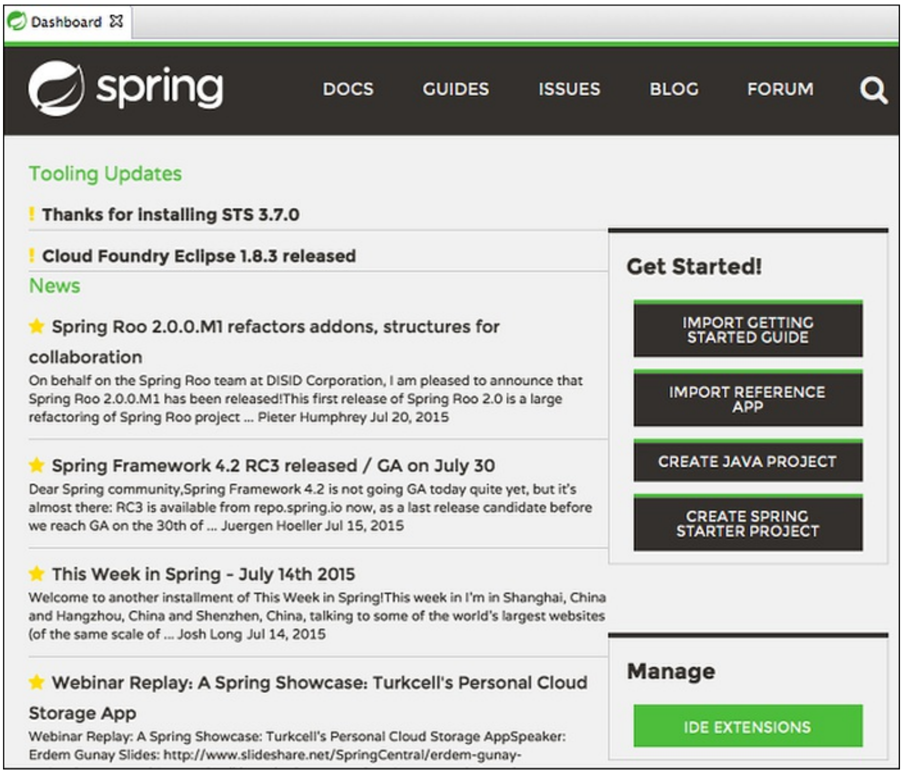


Install to STS from IDE Extensions Tab

Follow these instructions to install the Cloud Foundry Eclipse Plugin to Spring Tool Suite (STS) from the **IDE Extensions** tab.

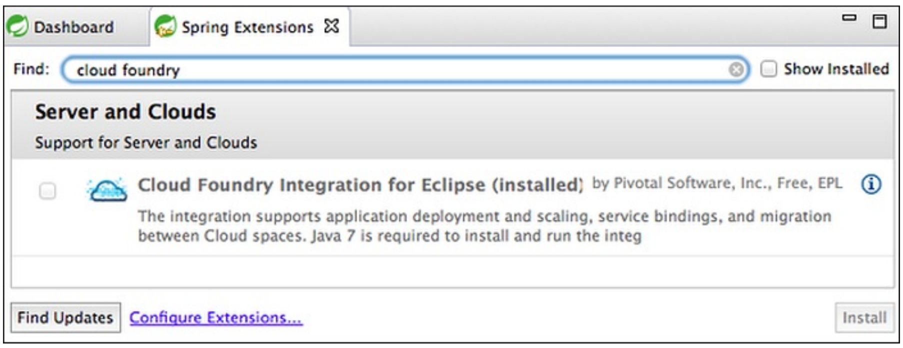
1. Start STS.

2. On the STS Dashboard, click **IDE Extensions**.

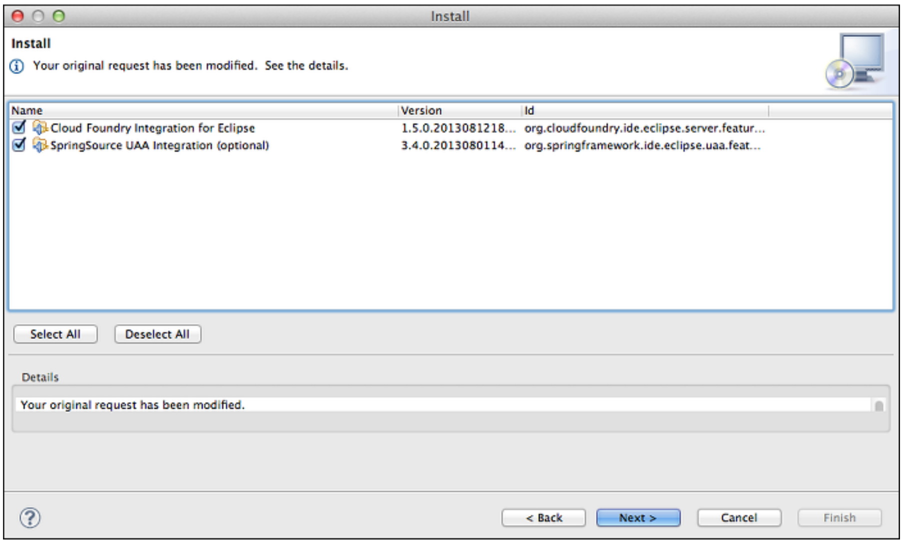


3. Enter “Cloud Foundry” in the **Find** field.

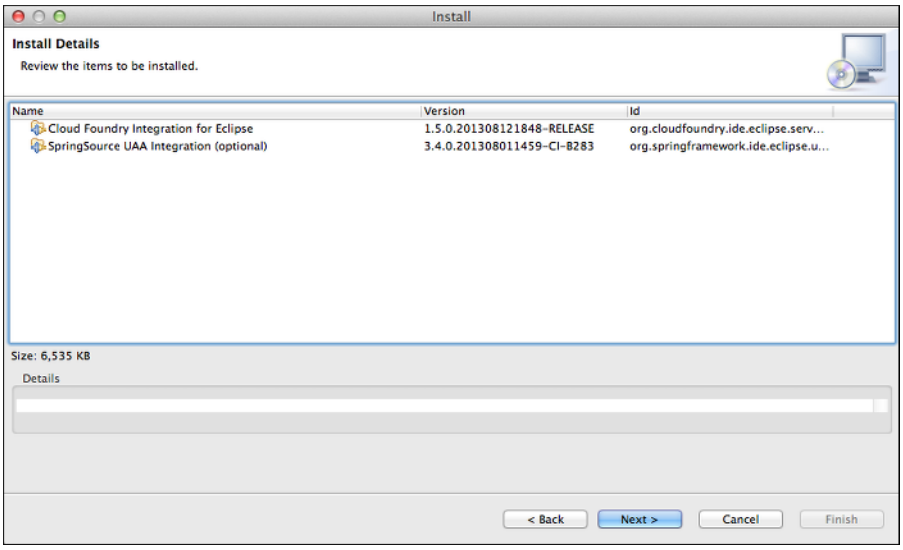
4. Select **Cloud Foundry Integration for Eclipse** and click **Install**.



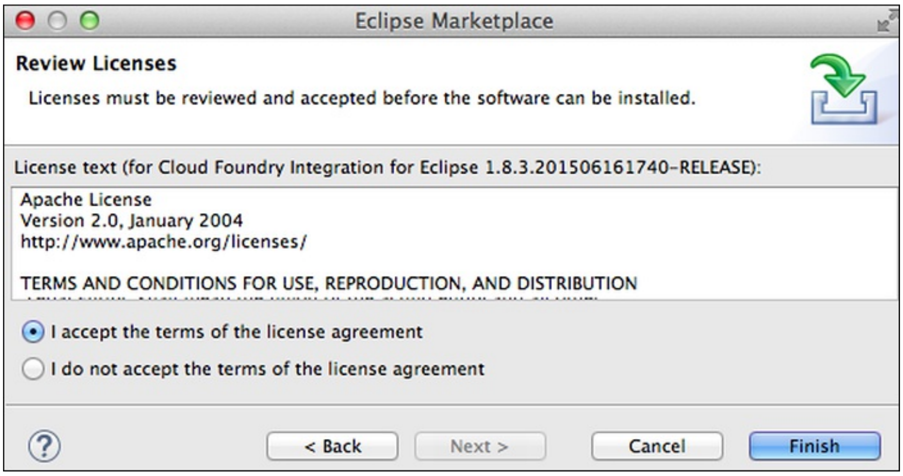
5. In the **Install** window, click **Next**.



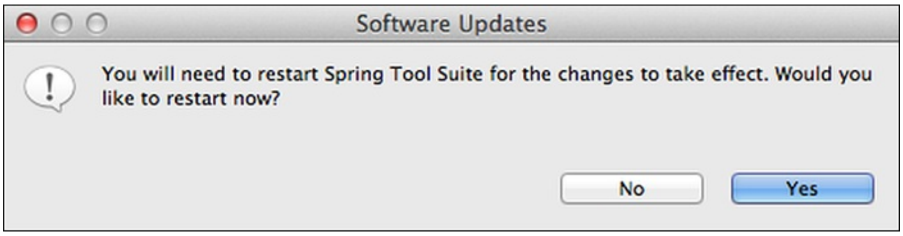
6. In the **Install Details** window, click **Next**.



7. The **Review Licenses** window appears. Select “I accept the terms of the license agreement” and click **Finish**.



8. The **Software Updates** window appears. Click **Yes** to restart Spring Tool Suite.



Install a Release Version Offline

If you need to install a release version of Cloud Foundry Eclipse Plugin in offline mode, you can download a release update site zip file and transfer it to the offline environment.

To install a Release Version offline, follow the steps below on a computer running Eclipse or Spring Tool Suite (STS).

1. Browse to https://github.com/cloudfoundry/eclipse-integration-cloudfoundry/blob/master/updatesites.md and download a release update site zip file.
2. In Eclipse or STS, select **Install New Software** from the **Help** menu.
3. In the **Available Software** window, to the right of the **Work with** field, click **Add**.
4. In the **Add Repository** window, enter `Cloud Foundry Integration` or a name of your choice for the repository. Click **Archive**.
5. In the **Open** window, browse to the location of the update site zip file and click **Open**.
6. In the **Add Repository** window, click **OK**.
7. In the **Available Software** window, select **Core/Cloud Foundry Integration** and, optionally, **Resources/Cloud Foundry Integration**. Click **Next**.
8. In the **Review Licenses** window, select “I accept the terms of the license agreement” and click **Finish**.

Install from a Local Build

If you need to install the Cloud Foundry Eclipse Plugin from a local build, rather than from a release version, you can download and build the source, create a repository and copy it to the target machine, then install from the copied repository.

1. Obtain the plugin source from GitHub in one of the following ways:
   - Download archived source code for released versions of the plugin from https://github.com/SpringSource/eclipse-integration-cloudfoundry/releases
   - Clone the project repository:

     ```
     $ git clone https://github.com/SpringSource/eclipse-integration-cloudfoundry
     ```

2. Unzip the downloaded archive. In a terminal, run the following command:

   ```
   $ mvn -Pe37 package
   ```

3. Copy the `org.cloudfoundry.ide.eclipse.server.site/target/site` directory to the machine where you want to install the plugin.
4. On the machine where you want to install the plugin, launch Eclipse or Spring Tool Suite (STS).
5. Select **Install New Software** from the **Help** menu.
6. In the **Available Software** window, to the right of the **Work with** field, click **Add**.
7. In the **Add Repository** window, enter `Cloud Foundry Integration` or a name of your choice for the repository. Click **Local**.
8. In the **Open** window, browse to the `org.cloudfoundry.ide.eclipse.server.site/target/site` directory. Click **Open**.
9. In the **Add Repository** window, click **OK**.

10. In the **Available Software** window, select **Core/Cloud Foundry Integration** and, optionally, **Resources/Cloud Foundry Integration**. Click **Next**.

11. In the **Review Licenses** window, select “I accept the terms of the license agreement” and click **Finish**.

About the Plugin User Interface

The sections below describe the Cloud Foundry Eclipse plugin user interface. If you do not see the tabs described below, select the Pivotal Cloud Foundry server in the **Servers** view. To expose the Servers view, ensure that you are using the Java perspective, then select **Window > Show View > Other > Server > Servers**.

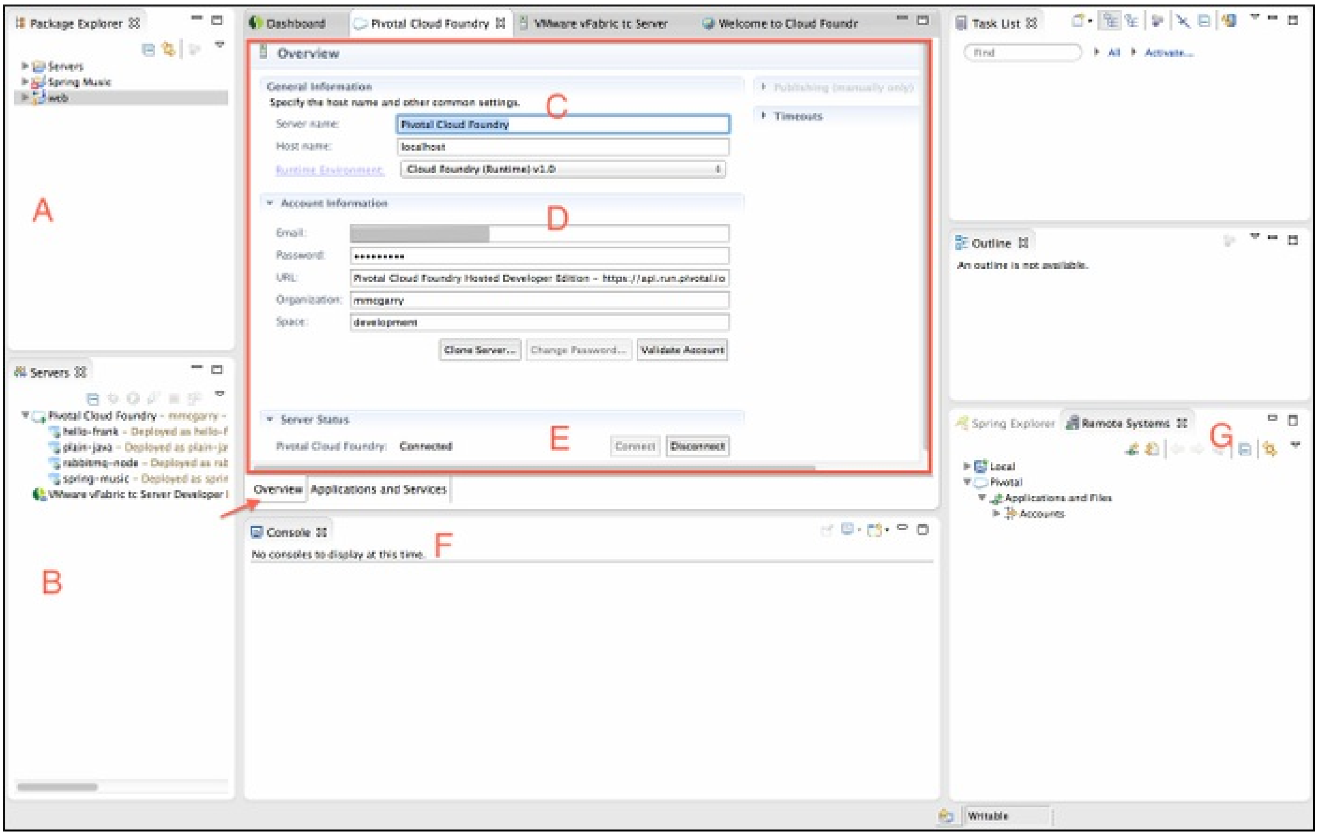
The Cloud Foundry editor, outlined in red in the screenshot below, is the primary plugin user interface. Some workflows involve interacting with standard elements of the Eclipse user interface, such as the **Project Explorer** and the **Console** and **Servers** views.

Note that the Cloud Foundry editor allows you to work with a single Cloud Foundry space. Each space is represented by a distinct server instance in the **Servers** view (B). Multiple editors, each targeting a different space, can be open simultaneously. However, only one editor targeting a particular Cloud Foundry server instance can be open at a time.

Overview Tab

The follow panes and views are present when the **Overview** tab is selected:

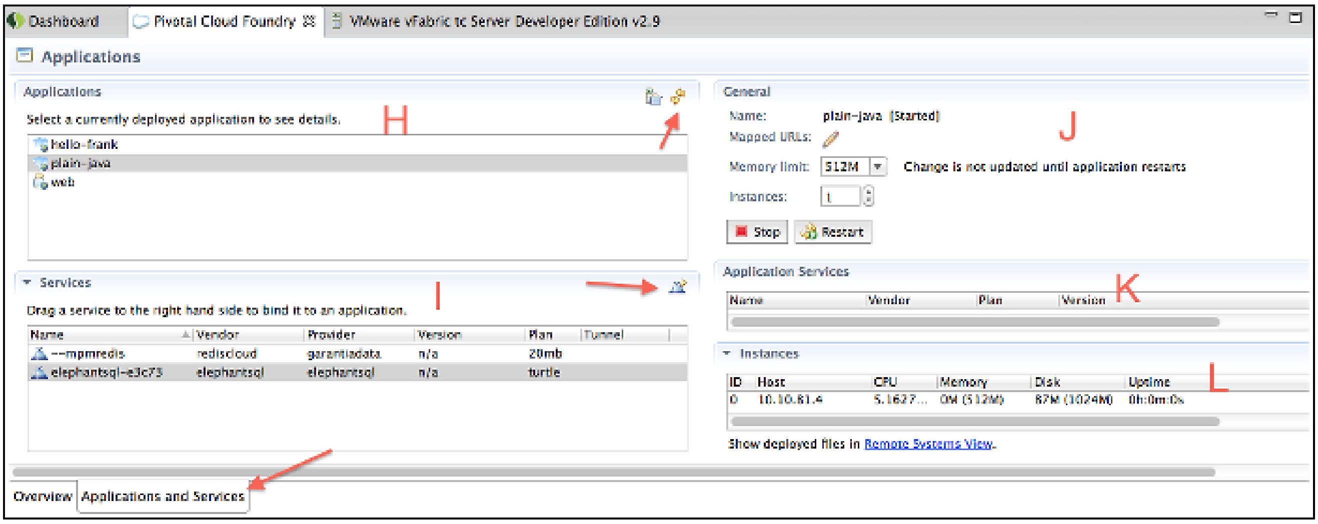
- A — The **Package Explorer** view lists the projects in the current workspace.
- B – The **Servers** view lists server instances configured in the current workspace. A server of type **Pivotal Cloud Foundry** represents a targeted space in a Cloud Foundry instance.
- C – The **General Information** pane.
- D – The **Account Information** pane lists your Cloud Foundry credentials and the target organization and space. The pane includes these controls:
  - **Clone Server** — Use to create additional Pivotal Cloud Foundry server instances. You must configure a server instance for each Cloud Foundry space that you wish to target. For more information, see Create Additional Server Instances.
  - **Change Password** — Use to change your Cloud Foundry password.
  - **Validate Account** — Use to verify your currently configured Cloud Foundry credentials.
- E – The **Server Status** pane shows whether or not you are connected to the target Cloud Foundry space, and the **Disconnect** and **Connect** controls.
- F – The **Console** view displays status messages when you perform an action such as deploying an application.
- G – The **Remote Systems** view allows you to view the contents of a file that is part of a deployed application. For more information, see View an Application File.



Applications and Services Tab

The follow panes are present when the **Applications and Services** tab is selected:

- H — The **Applications** pane lists the applications deployed to the target space.
- I — The **Services** pane lists the services provisioned in the targeted space.
- J — The **General** pane displays the following information for the application currently selected in the **Applications** pane:
    - **Name**
    - **Mapped URLs** – Lists URLs mapped to the application. You can click a URL to open a browser to the application within Eclipse or STS, and click the pencil icon to add or remove mapped URLs. See Manage Application URLs.
    - **Memory Limit** – The amount of memory allocated to the application. You can use the pull-down to change the memory limit.
    - **Instances** – The number of instances of the application that are deployed. You can use the pull-down to change number of instances.
    - **Start**, **Stop**, **Restart**, **Update and Restart** — The controls that appear depend on the current state of the application. The **Update and Restart** command will attempt an incremental push of only those application resources that have changed. It will not perform a full application push. See Push Application Changes below.
- K — The **Services** pane lists services that are bound to the application currently selected in the **Applications** pane. The icon in the upper right corner of the pane allows you to create a service, as described in Create a Service.

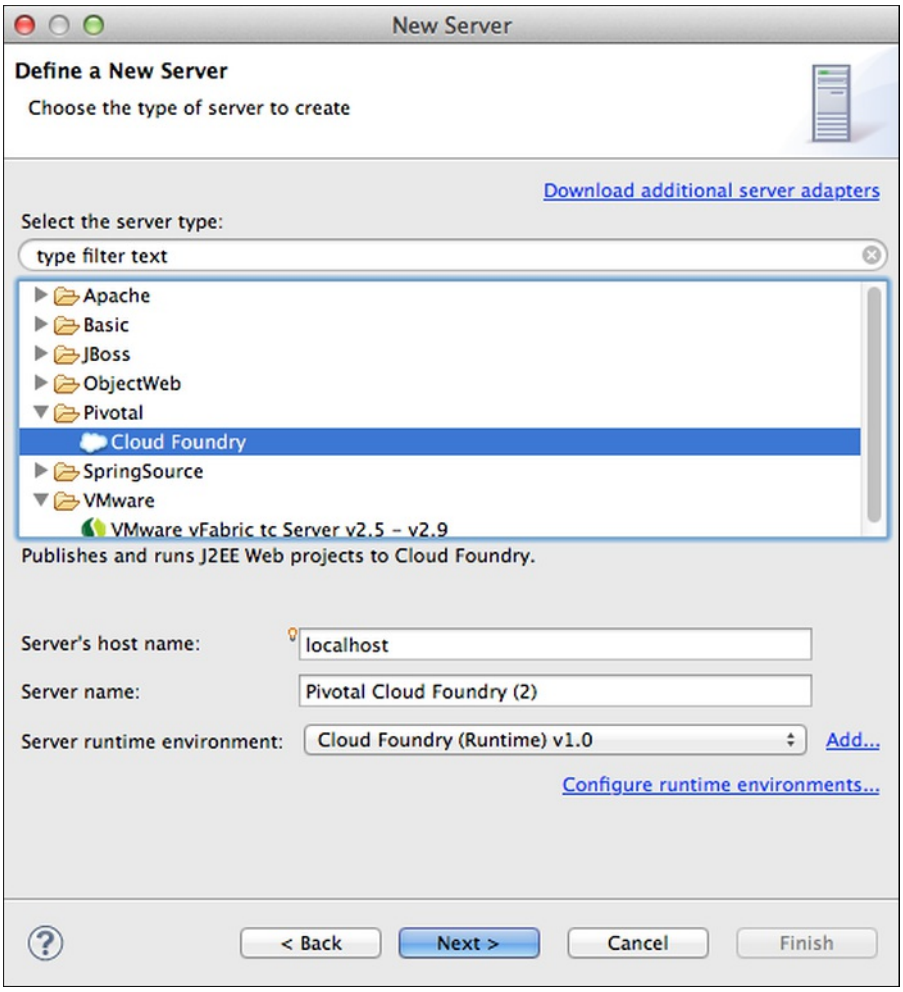


Create a Cloud Foundry Server

This section contains instructions for configuring a server resource that will represent a target Cloud Foundry space. You will create a server for each space in Cloud Foundry to which you will deploy applications. Once you create your first Cloud Foundry service instances using the instructions below, you can create additional instances using the Clone Server feature.

1. Right-click the **Servers** view and select **New > Server**.
2. In the **Define a New Server** window, expand the **Pivotal** folder, select **Cloud Foundry**, and click **Next**.

> **Note**: Do not modify default values for **Server host name** or **Server Runtime Environment**. These fields are not used

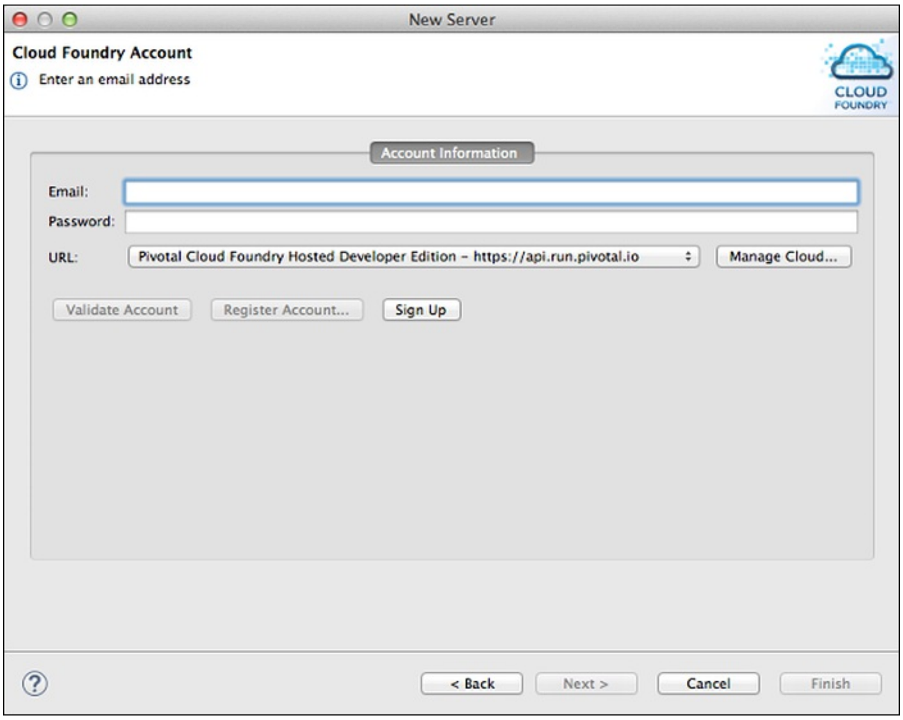


3. In the **Cloud Foundry Account** window, if you already have a Pivotal Cloud Foundry Hosted Developer Edition account, enter your email account and password credentials and click **Validate Account**.

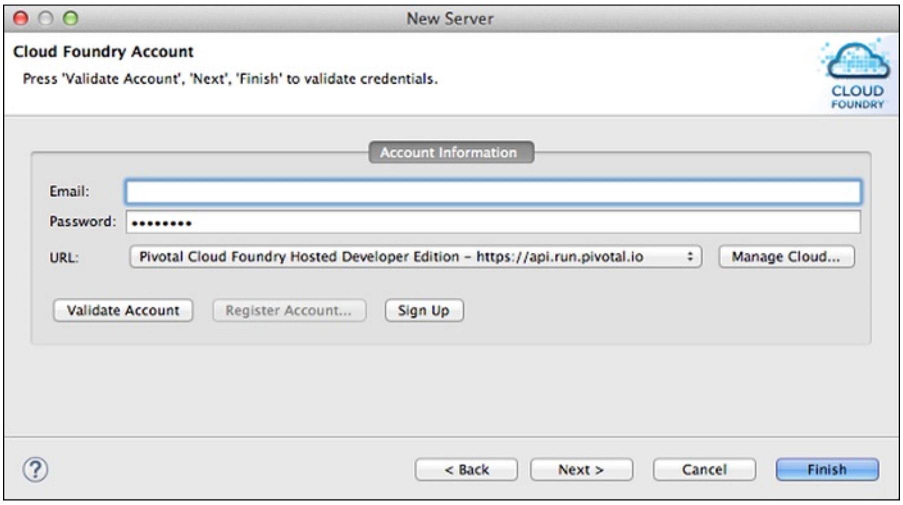
> **Note**: By default, the **URL** field points to the Pivotal Cloud Foundry Hosted Developer Edition URL, `https://api.run.pivotal.io`. If you have a PAS account, refer to the Logging in to Apps Manager topic to determine your PAS URL. Click **Manage Cloud...** to add this URL to your Cloud Foundry account. Validate the account and continue through the wizard.

If you do not have a Cloud Foundry account and want to register a new Pivotal Cloud Foundry Hosted Developer Edition account, click **Sign Up**. After you create the account, you can complete this procedure.

> **Note**: The **Register Account** button is inactive.

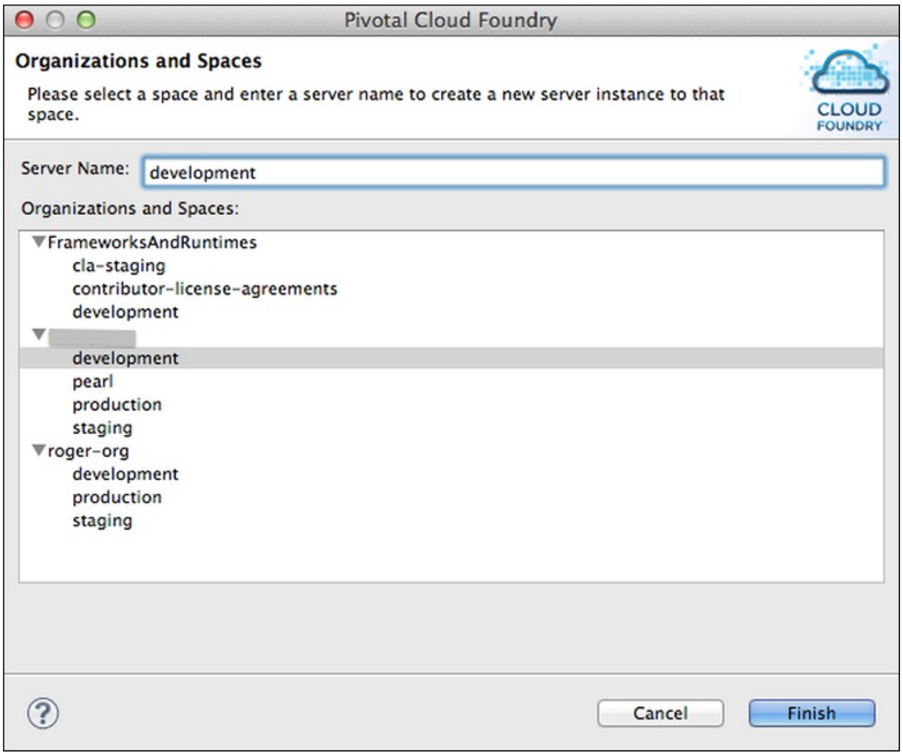


4. The **Cloud Foundry Account** window is refreshed and displays a message indicating whether or not your credentials were valid. Click **Next**.



5. In the **Organizations and Spaces** window, select the space that you want to target, and click **Finish**.

> **Note**: If you do not select a space, the server will be configured to connect to the default space, which is the first encountered in a list of your spaces.



6. Once you have successfully configured the Pivotal Cloud Foundry server, it will appear in the **Servers** view of the Eclipse or STS user interface. To familiarize yourself with the plugin user interface, see About the Plugin User Interface. Following this step, proceed to Deploy an Application.

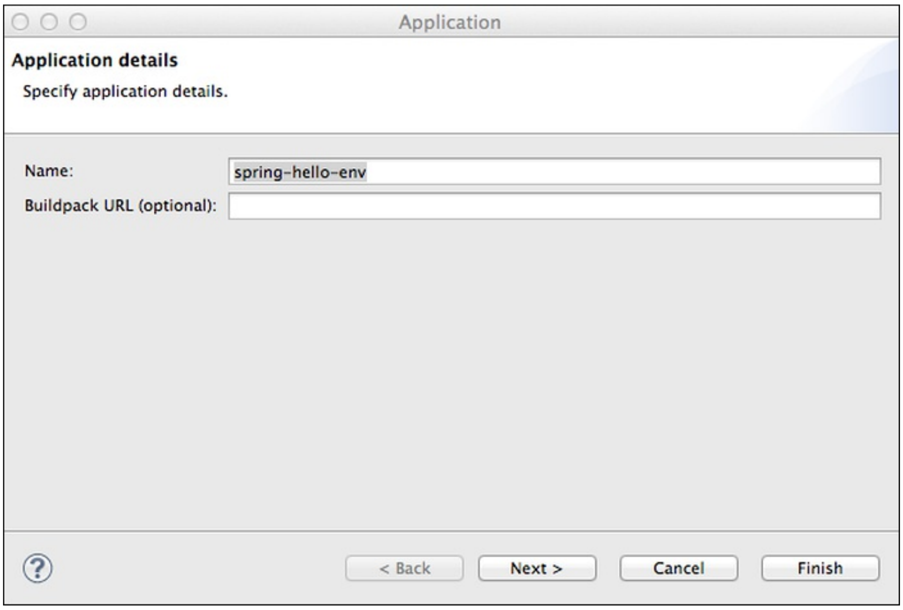
Deploy an Application

To deploy an application to Cloud Foundry using the plugin:

1. To initiate deployment either:
   - Drag the application from the **Package Explorer** view onto the Pivotal Cloud Foundry server in the **Servers** view, or
   - Right-click the Pivotal Cloud Foundry server in the **Servers** view, select **Add and Remove** from the server context menu, and move the application from the **Available** to the **Configured** column.

2. In the **Application Details** window:
   - By default, the **Name** field is populated with the application project name. You can enter a different name. The name is assigned to the deployed application, but does not rename the project.
   - If you want to use an external buildpack to stage the application, enter the URL of the buildpack.

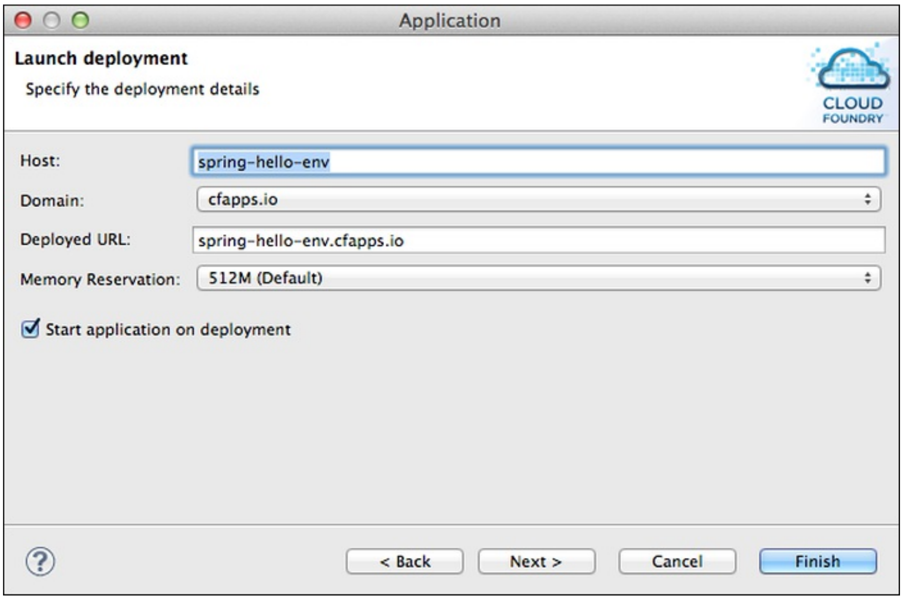
   You can deploy the application without further configuration by clicking **Finish**. Note that because the application default values may take a second or two to load, the **Finish** button might not be enabled immediately. A progress indicator will indicate when the application default values have been loaded, and the **Finish** button will be enabled.
   Click **Next** to continue.



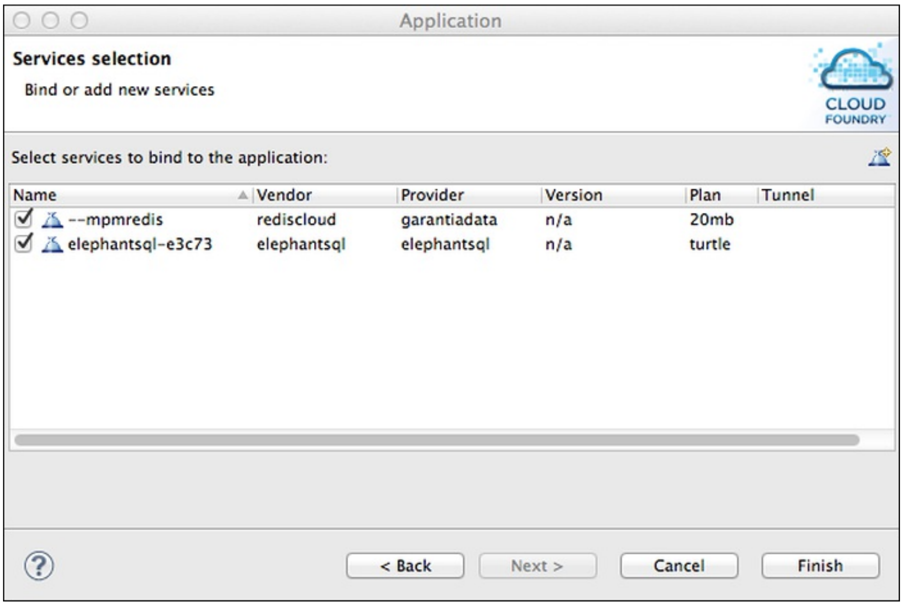
3. In the **Launch Deployment** window:
   Host — By default, contains the name of the application. You can enter a different value if desired. If you push the same application to multiple spaces in the same organization, you must assign a unique **Host** to each.
   Domain — Contains the default domain. If you have mapped custom domains to the target space, they appear in the pull-down list.

   > **Note:** This version of the Cloud Foundry Eclipse plugin does not provide a mechanism for mapping a custom domain to a space. You must use the `cf map domain` command to do so.

   Deployed URL — By default, contains the value of the **Host** and **Domain** fields, separated by a period (.) character.
   Memory Reservation — Select the amount of memory to allocate to the application from the pull-down list.
   Start application on deployment — If you do not want the application to be started on deployment, uncheck the box.



4. The **Services Selection** window lists services provisioned in the target space. Checkmark the services, if any, that you want to bind to the application, and click **Finish**. You can bind services to the application after deployment, as described in Bind and Unbind Services.



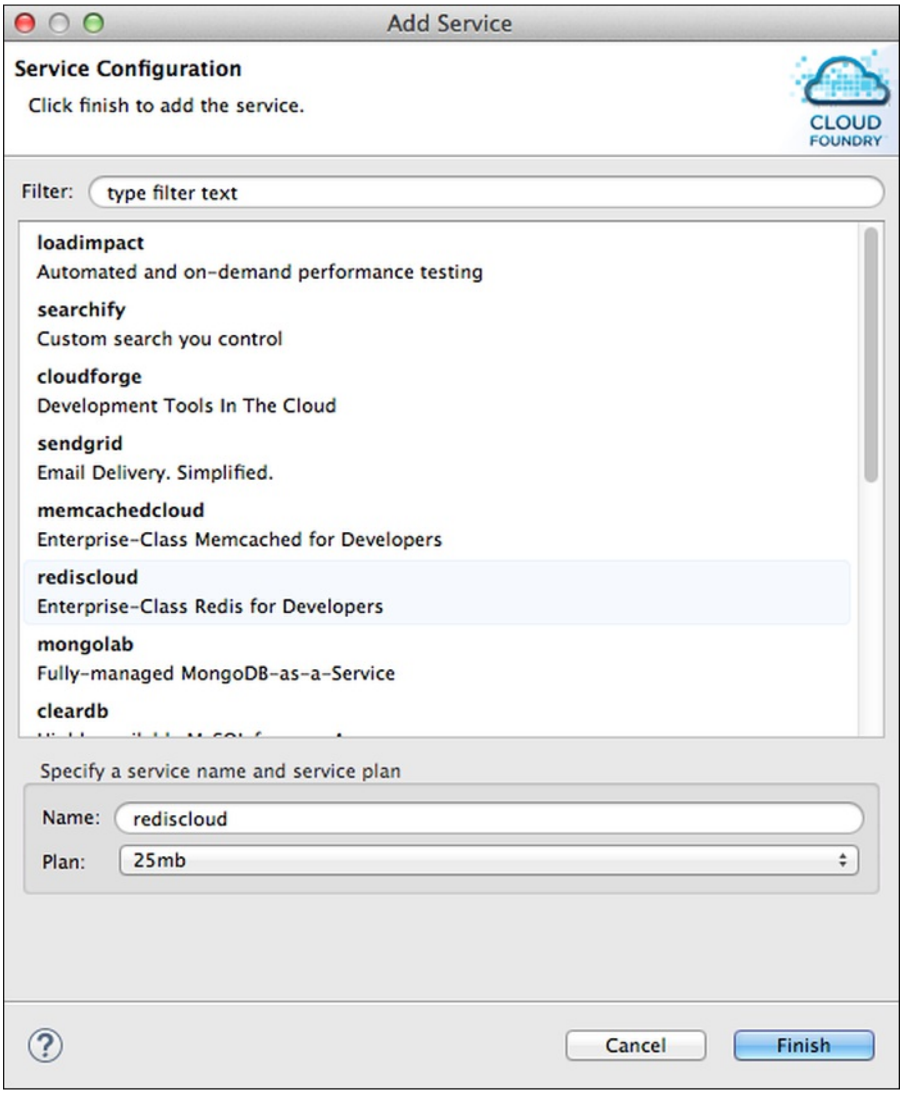
As the deployment proceeds, progress messages appear in the **Console** view. When deployment is complete, the application is listed in the **Applications** pane.

Create a Service

Before you can bind a service to an application, you must create it.

To create a service:

1. Select the **Applications and Services** tab.
2. Click the icon in the upper right corner of the **Services** pane.
3. In the **Service Configuration** window, enter a text pattern to **Filter** for a service. Matches are made against both service name and description.
4. Select a service from the **Service List**. The list automatically updates based on the filter text.
5. Enter a **Name** for the service and select a service **Plan** from the drop-down list.



6. Click **Finish**. The new service appears in the **Services** pane.

Bind and Unbind Services

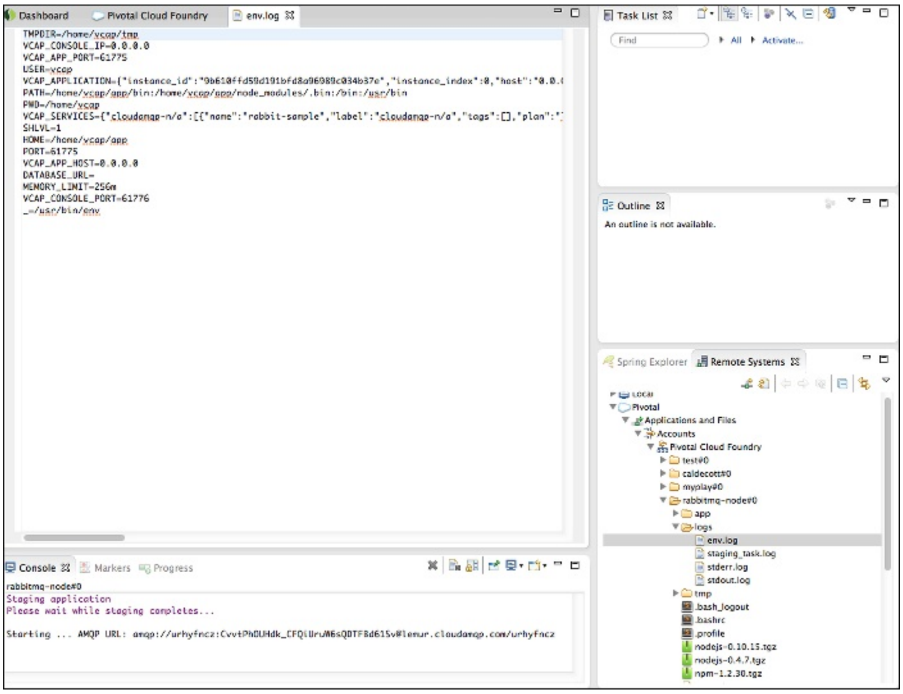
You can bind a service to an application when you deploy it. To bind a service to an application that is already deployed, drag the service from the **Services** pane to the **Application Services** pane. (See the area labelled "G" in the screenshot in the Applications and Services above.)

To unbind a service, right-click the service in the **Application Services** pane, and select **Unbind from Application**.

View an Application File

You can view the contents of a file in a deployed application by selecting it the **Remote Systems View**. (See the areas labelled "I" and "J" in the screenshot in the Applications and Services Tab above.)

1. If the **Remote Systems View** is not visible:
   - Select the **Applications and Services** tab.
   - Select the application of interest from the **Applications** pane.
   - In the **Instances** pane, click the **Remote Systems View** link.
2. In the **Remote Systems View**, browse to the application and application file of interest, and double-click the file. A new tab appears in the editor area with the contents of the selected file.



Undeploy an Application

To undeploy an application, right click the application in either the **Servers** or the **Applications** pane and click **Remove**.

Scale an Application

You can change the memory allocation for an application and the number of instances deployed in the **General** pane when the **Applications and Services** tab is selected. Use the **Memory Limit** and **Instances** selector lists.

Although the scaling can be performed while the application is running, if the scaling has not taken effect, restart the application. If necessary, manually refresh the application statistics by clicking **Refresh** in the top, right corner of the “Applications” pane, labelled “H” in the screenshot in Applications and Services Tab.

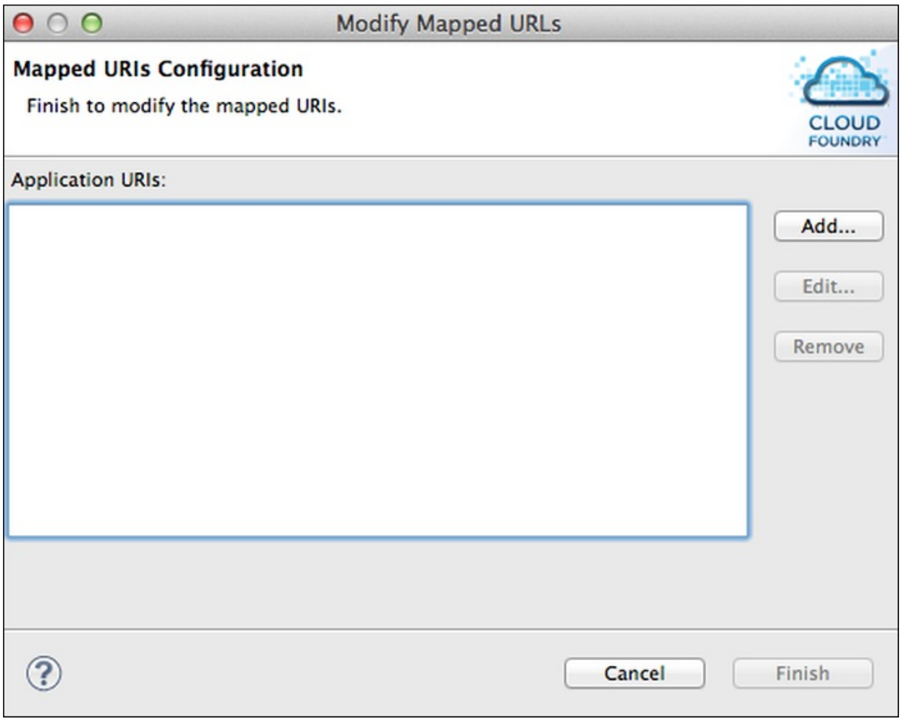
Push Application Changes

The Cloud Foundry editor supports these application operations:

- **Start** and **Stop** — When you **Start** an application, the plugin pushes all application files to the Cloud Foundry instance before starting the application, regardless of whether there are changes to the files or not.
- **Restart** — When you **Restart** a deployed application, the plugin does not push any resources to the Cloud Foundry instance.
- **Update and Restart** — When you run this command, the plugin pushes only the changes that were made to the application since last update, not the entire application. This is useful for performing incremental updates to large applications.

Manage Application URLs

You add, edit, and remove URLs mapped to the currently selected application in the **General** pane when the **Applications and Services** tab is selected. Click the pencil icon to display the **Mapped URIs Configuration** window.



Information in the Console View

When you start, restart, or update and restart an application, application output will generally be streamed to the **Console** view (labelled "F" in the screenshot in [Overview Tab](#)). The information shown in the **Console** view for a running application instance includes staging information, and the application's `std.out` and `std.error` logs.

If multiple instances of the application are running, only the output of the first instance appears in the **Console** view. To view the output of another running instance, or to refresh the output that is currently displayed:

1. In the **Applications and Services** tab, select the deployed application in the **Applications** pane.
2. Click **Refresh** on the top right corner of the **Applications** pane.
3. In the **Instances** pane, wait for the application instances to be updated.
4. Once non-zero health is shown for an application instance, right-click on that instance to open the context menu and select **Show Console**.

Clone a Cloud Foundry Server Instance

Each space in Cloud Foundry to which you want to deploy applications must be represented by a Cloud Foundry server instance in the **Servers** view. After you have created a Cloud Foundry server instance, as described in [Create a Cloud Foundry Server](#), you can clone it to create another.

Follow the step below to clone a server:

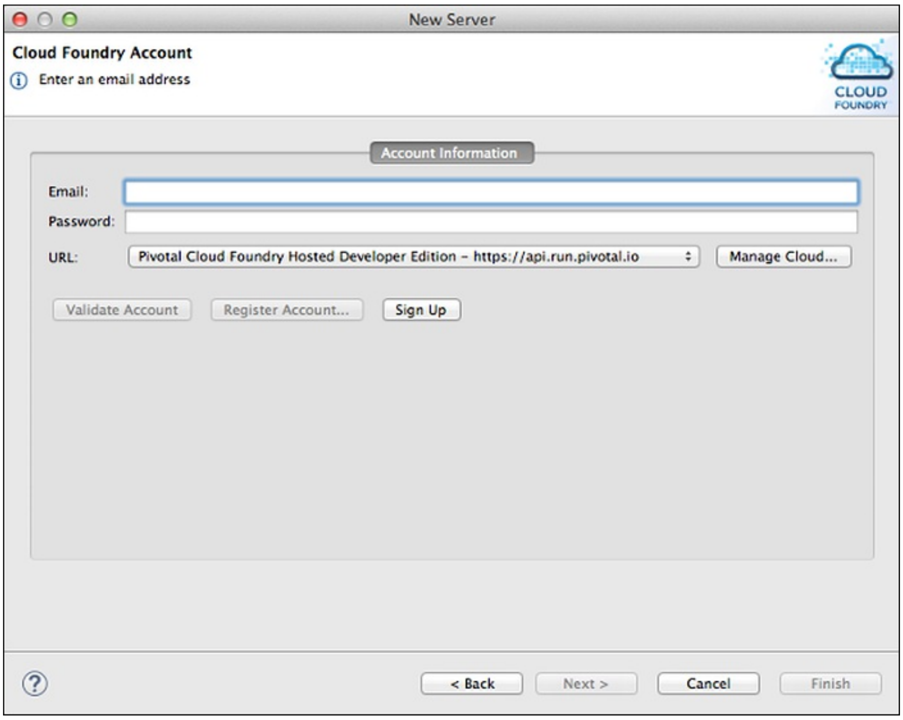
1. Perform one of the following actions:
   - In the Cloud Foundry server instance editor "Overview" tab, click **Clone Server**.
   - Right-click a Cloud Foundry server instance in the **Servers** view, and select **Clone Server** from the context menu.
2. In the **Organizations and Spaces** window, select the space that you want to target.
3. The name field will be filled with the name of the space that you selected. If desired, edit the server name before clicking finish **Finish**.

Add a Cloud Foundry Instance URL

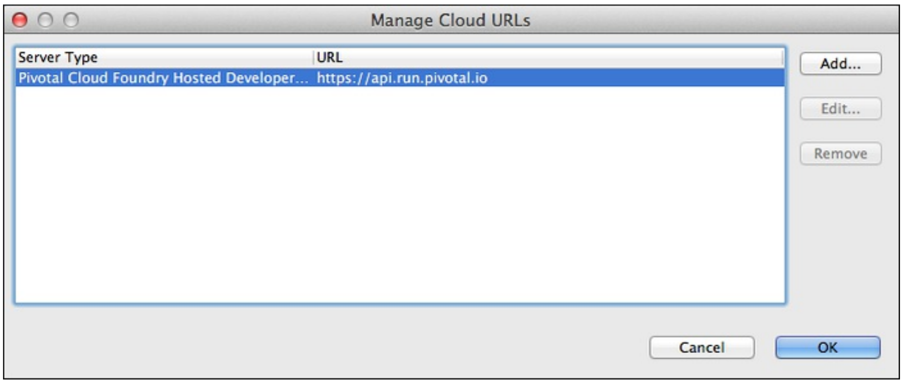
You can configure the plugin to work with any Cloud Foundry instances to which you have access. To do so:

1. Perform steps 1 and 2 of Create a Cloud Foundry Server.

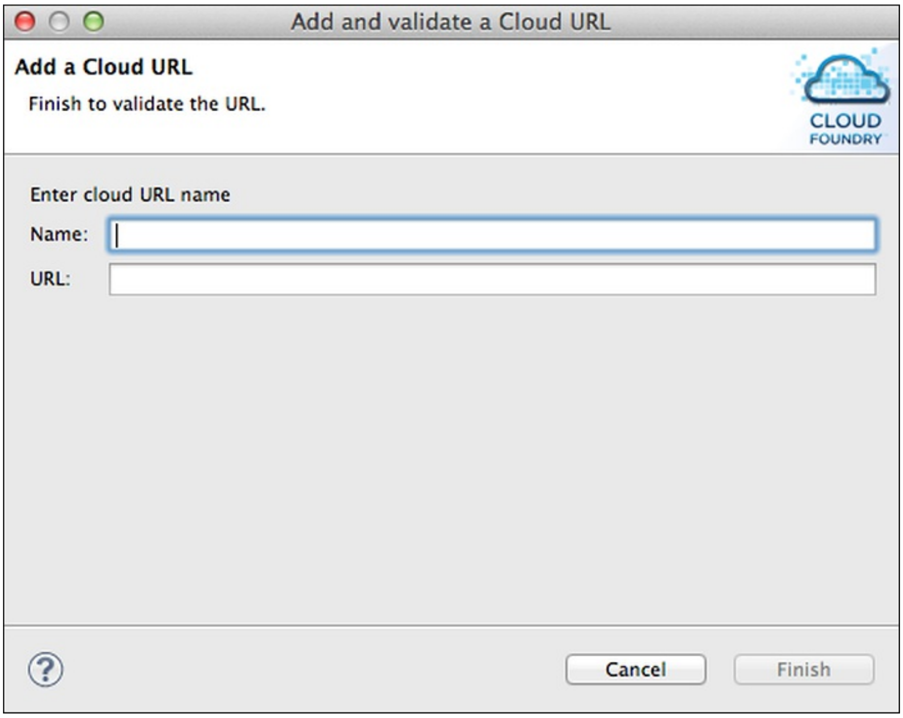
2. In the **Cloud Foundry Account** window, enter the email account and password that you use to log on to the target instance, then click **Manage Cloud URLs**



3. In the **Manage Cloud URLs** window, click **Add**.



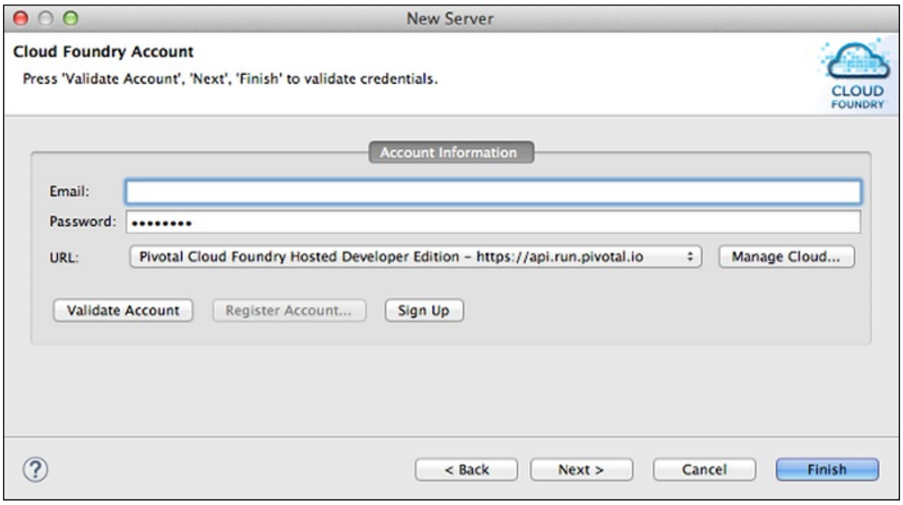
4. In the **Add a Cloud URL** window, enter the name and URL of the target cloud instance and click **Finish**.



5. The new cloud instance should appear in the list on the **Manage Cloud URLs** window. Click **OK** to proceed.

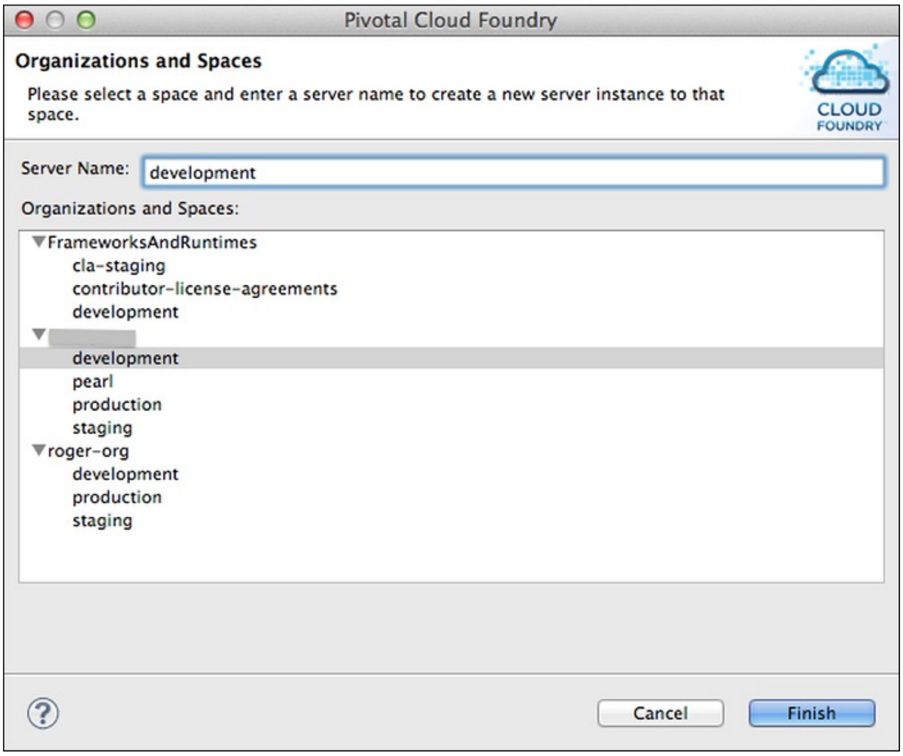
6. In the **Cloud Foundry Account** window, click **Validate Account**.

7. The **Cloud Foundry Account** window is refreshed and displays a message indicating whether or not your credentials were valid. Click **Next**.



8. In the **Organizations and Spaces** window, select the space that you want to target, and click **Finish**.

> **Note:** If you do not select a space, the server will be configured to connect to the default space, which is the first encountered in a list of your spaces.



9. Once you have successfully configured the Pivotal Cloud Foundry server, it will appear in the **Servers** view of the Eclipse or STS user interface. To familiarize yourself with the plugin user interface, see About the Plugin User Interface. Following this step, proceed to Deploy an Application.

Cloud Foundry Java Client Library

Page last updated:

Introduction

This is a guide to using the Cloud Foundry Java Client Library to manage an account on a Cloud Foundry instance.

Note: The 1.1.x versions of the Cloud Foundry Java Client Library work with apps using Spring 4.x, and the 1.0.x versions of the Cloud Foundry Java Client Library work with apps using Spring 3.x. Both versions are available in the source repository on GitHub.

Adding the Library

Visit the Cloud Foundry Java Client Library GitHub page to obtain the correct components.

Most projects need two dependencies: the Operations API and an implementation of the Client API. Refer to the following sections for more information about how to add the Cloud Foundry Java Client Library as dependencies to a Maven or Gradle project.

Maven

Add the `cloudfoundry-client-reactor` dependency (formerly known as `cloudfoundry-client-spring`) to your `pom.xml` as follows:

```
<dependencies>
    <dependency>
        <groupId>org.cloudfoundry</groupId>
        <artifactId>cloudfoundry-client-reactor</artifactId>
        <version>2.0.0.BUILD-SNAPSHOT</version>
    </dependency>
    <dependency>
        <groupId>org.cloudfoundry</groupId>
        <artifactId>cloudfoundry-operations</artifactId>
        <version>2.0.0.BUILD-SNAPSHOT</version>
    </dependency>
    <dependency>
        <groupId>io.projectreactor</groupId>
        <artifactId>reactor-core</artifactId>
        <version>2.5.0.BUILD-SNAPSHOT</version>
    </dependency>
    <dependency>
        <groupId>io.projectreactor</groupId>
        <artifactId>reactor-netty</artifactId>
        <version>2.5.0.BUILD-SNAPSHOT</version>
    </dependency>
    ...
</dependencies>
```

The artifacts can be found in the Spring release and snapshot repositories:

```
<repositories>
    <repository>
        <id>spring-releases</id>
        <name>Spring Releases</name>
        <url>http://repo.spring.io/release</url>
    </repository>
    ...
</repositories>
```

```
<repositories>
    <repository>
        <id>spring-snapshots</id>
        <name>Spring Snapshots</name>
        <url>http://repo.spring.io/snapshot</url>
        <snapshots>
            <enabled>true</enabled>
        </snapshots>
    </repository>
    ...
</repositories>
```

Gradle

Add the `cloudfoundry-client-reactor` dependency to your `build.gradle` file as follows:

```
dependencies {
    compile 'org.cloudfoundry:cloudfoundry-client-reactor:2.0.0.BUILD-SNAPSHOT'
    compile 'org.cloudfoundry:cloudfoundry-operations:2.0.0.BUILD-SNAPSHOT'
    compile 'io.projectreactor:reactor-core:2.5.0.BUILD-SNAPSHOT'
    compile 'io.projectreactor:reactor-netty:2.5.0.BUILD-SNAPSHOT'
    ...
}
```

The artifacts can be found in the Spring release and snapshot repositories:

```
repositories {
    maven { url 'http://repo.spring.io/release' }
    ...
}
```

```
repositories {
    maven { url 'http://repo.spring.io/snapshot' }
    ...
}
```

Sample Code

The following is a very simple sample application that connects to a Cloud Foundry instance, logs in, and displays some information about the Cloud Foundry account. When running the program, provide the Cloud Foundry target API endpoint, along with a valid user name and password as command-line parameters.

```java
import org.cloudfoundry.client.lib.CloudCredentials;
import org.cloudfoundry.client.lib.CloudFoundryClient;
import org.cloudfoundry.client.lib.domain.CloudApplication;
import org.cloudfoundry.client.lib.domain.CloudService;
import org.cloudfoundry.client.lib.domain.CloudSpace;

import java.net.MalformedURLException;
import java.net.URI;
import java.net.URL;

public final class JavaSample {

  public static void main(String[] args) {
    String target = args[0];
    String user = args[1];
    String password = args[2];

    CloudCredentials credentials = new CloudCredentials(user, password);
    CloudFoundryClient client = new CloudFoundryClient(credentials, getTargetURL(target));
    client.login();

    System.out.printf("%nSpaces:%n");
    for (CloudSpace space : client.getSpaces()) {
      System.out.printf("  %s\t(%s)%n", space.getName(), space.getOrganization().getName());
    }

    System.out.printf("%nApplications:%n");
    for (CloudApplication application : client.getApplications()) {
      System.out.printf("  %s%n", application.getName());
    }

    System.out.printf("%nServices%n");
    for (CloudService service : client.getServices()) {
      System.out.printf("  %s\t(%s)%n", service.getName(), service.getLabel());
    }
  }

  private static URL getTargetURL(String target) {
    try {
      return URI.create(target).toURL();
    } catch (MalformedURLException e) {
      throw new RuntimeException("The target URL is not valid: " + e.getMessage());
    }
  }

}
```

For more details about the Cloud Foundry Java Client Library, visit the source repository in GitHub. The domain package shows the objects that you can query and inspect.

.NET Core Buildpack

Page last updated:

This topic describes how to push Cloud Foundry apps using the .NET Core buildpack. You can find supported ASP.NET Core versions in the .NET Core buildpack release notes.

> **Note**: The .NET Core buildpack only works with ASP.NET Core. For apps which are not based on this toolchain, refer to the legacy .NET buildpack.

Push an App

Cloud Foundry automatically uses the .NET Core buildpack when one or more of the following conditions are met:

- The pushed app contains one or more `*.csproj` or `*.fsproj` files.
- The pushed app contains one or more `project.json` files.
- The app is pushed from the output directory of the `dotnet publish` command.

If your Cloud Foundry deployment does not have the .NET Core buildpack installed or the installed version is out of date, push your app with the `-b` option to specify the buildpack:

```
$ cf push MY-APP -b https://github.com/cloudfoundry/dotnet-core-buildpack.git
```

Specify any non-default package sources in the `NuGet.Config` file.

For a basic example, see this Hello World sample.

.NET Core SDKs

The first several releases of the .NET Core SDKs used `project.json` files for project build configuration. The current release of the .NET Core SDK uses MSBuild as its build tool, which uses `*.csproj` and `*.fsproj` files for configuration.

Currently, the .NET Core buildpack includes both types of SDKs. If the pushed app contains a `global.json` file, the buildpack installs the version specified by that file. Otherwise, the buildpack chooses which SDK to install as follows:

1. If the app only contains `project.json` files, the buildpack installs the latest SDK that supports this configuration.
2. If the app only contains `*.csproj` and `*.fsproj` files, the buildpack installs the latest SDK that uses MSBuild.
3. If the app contains both file types without a `global.json`, the buildpack throws an error, as it cannot determine the proper SDK to install.

> **Note**: Microsoft has removed support for project.json from the .NET Core SDK tools. Consequently, support for `project.json` apps in the buildpack will soon be deprecated.

Configure the Listen Port

For your .NET Core app to work on Cloud Foundry, you must modify the `Main` method to configure the app to listen on the port specified by the `$PORT` environment variable, which Cloud Foundry sets automatically.

1. Open the file that contains your `Main` method.
2. Add a `using` statement to the top of the file.

```
using Microsoft.Extensions.Configuration;
```

3. Add the following lines before the line `var host = new WebHostBuilder()`:

```
var config = new ConfigurationBuilder()
  .AddCommandLine(args)
  .Build();
```

4. Add the following line after `.UseKestrel()`:

```
.UseConfiguration(config)
```

   This allows the buildpack to pass the correct port from `$PORT` to the app when running the initial startup command.

5. Add `Microsoft.Extensions.Configuration.CommandLine` as a dependency using one of the following:
   - `project.json`:

     ```
     "Microsoft.Extensions.Configuration.CommandLine": "VERSION",
     ```

   - `*.csproj`:

     ```
     <PackageReference Include="Microsoft.Extensions.Configuration.CommandLine">
       <Version>VERSION</Version>
     </PackageReference>
     ```

   where VERSION is the version of the package to use. Valid versions can be found on `https://www.nuget.org`.

6. If your app requires any other files at runtime, such as JSON configuration files, add them to the `include` section of `copyToOutput`.

7. Save your changes.

With these changes, the `dotnet run` command copies your app `Views` to the build output, where the .NET CLI can find them. Refer to the following example `Main` method:

```
public static void Main(string[] args)
{
  var config = new ConfigurationBuilder()
    .AddCommandLine(args)
    .Build();

  var host = new WebHostBuilder()
    .UseKestrel()
    .UseConfiguration(config)
    .UseContentRoot(Directory.GetCurrentDirectory())
    .UseStartup<Startup>()
    .Build();
  host.Run();
}
```

Note: The samples provided in the cli-samples repository and the templates provided by Visual Studio and Yeoman work with this buildpack after you have followed the steps above.

Specify a .NET Framework Version

Lock the .NET Framework to a minor version. Do not lock to a patch version, because buildpacks contain only the two most recent patch versions of each minor version.

To lock the .NET Framework version in a `.csproj` app:

```
<ItemGroup>
  <PackageReference Include="Microsoft.NETCore.App">
    <Version>1.1.*</Version>
  </PackageReference>
</ItemGroup>
```

To lock the .NET Framework version in a `project.json` app:

```
"dependencies": {
  "Microsoft.NETCore.App": {
    "type": "platform",
    "version": "1.0.*"
  }
}
```

Deploy Apps with Multiple Projects

To deploy an app that contains multiple projects, you must specify a main project for the buildpack to run. Create a `.deployment` file in the root folder of the app which sets the path to the main project as follows:

```
[config]
project = <main project>
```

1. If the app uses `project.json`, set `project` to the directory containing the `project.json` file of the main project.
2. If the app uses MSBuild, set `project` to the `*.csproj` or `*.fsproj` file of the main project.

For example, if an app using MSBuild contains three projects in the `src` folder, the main project `MyApp.Web`, `MyApp.DAL`, and `MyApp.Services`, format the `.deployment` file as follows:

```
[config]
project = src/MyApp.Web/MyApp.Web.csproj
```

In this example, the buildpack automatically compiles the `MyApp.DAL` and `MyApp.Services` projects if the `MyApp.Web.csproj` file of the main project lists them as dependencies, `MyApp.Web`. The buildpack attempts to execute the main project with `dotnet run -p src/MyApp.Web/MyApp.Web.csproj`.

Push an App in a Disconnected Environment

For offline use, you can cache the binaries in `manifest.yml` with the buildpack.

You can push apps with their other dependencies following these steps:

1. Publish the app by running `dotnet publish -r ubuntu.14.04-x64`.

   > **Note**: For this publish command to work, modify your app code so the .NET CLI publishes it as a self-contained app. For more information, see .NET Core Application Deployment.

2. Navigate to the `bin/<Debug|Release>/<framework>/<runtime>/publish` directory. Or, if your app uses a `manifest.yml`, specify a path to the publish output folder. This allows you to push your app from any directory.
3. Push your app.

Disabling the NuGet Package Cache

You may need to disable NuGet package caching, or clear NuGet packages cached in the staging environment, in one of the following scenarios:

- Your app fails to stage because it runs out of space, exceeding the maximum allowable disk quota.
- You have added pre-release packages to test a new feature, then decided to revert back to the main NuGet feed. You may need to remove the packages you changed from the cache to avoid conflicts.

Disabling NuGet caching both clears any existing NuGet dependencies from the staging cache and prevents the buildpack from adding NuGet dependencies to the staging cache.

To disable NuGet package caching, set the `CACHE_NUGET_PACKAGES` environment variable to `false`. If the variable is not set, or set to a different value, there is no change. Perform one of the following procedures to set `CACHE_NUGET_PACKAGES` to `false`:

- Locate your app manifest, `manifest.yml`, and set the `CACHE_NUGET_PACKAGES` environment variable, following the format of the example below:

```
---
applications:
- name: sample-aspnetcore-app
  memory: 512M
  env:
    CACHE_NUGET_PACKAGES: false
```

- Use `cf set-env` to set the `CACHE_NUGET_PACKAGES` environment variable on the command line:

```
$ cf set-env YOUR-APP CACHE_NUGET_PACKAGES false
```

See the Environment Variables section of the *Deploying with Application Manifests* topic for more information.

Add Custom Libraries

If your app requires external shared libraries that are not provided by the rootfs or the buildpack, you must place the libraries in an `ld_library_path` directory at the app root.

Note: You must keep these libraries up-to-date. They do not update automatically.

The .NET Core buildpack automatically adds the directory `<app-root>/ld_library_path` to `LD_LIBRARY_PATH` so that your app can access these libraries at runtime.

Legacy .NET Buildpack

A legacy .NET buildpack exists, built by the Cloud Foundry community. This buildpack addresses apps not based on .NET Core. However, it requires you to write and compile your apps using Mono, for example from Xamarin Studio, MonoDevelop, or xbuild.

Node.js Buildpack

Page last updated:

Use the Node.js buildpack with Node or JavaScript apps.

You must install the Cloud Foundry Command Line Interface tool (cf CLI) to run some of the commands listed in this topic.

Push Node.js Apps

Cloud Foundry automatically uses the Node.js buildpack if it detects a `package.json` file in the root directory of your project.

The `-b` option lets you specify a buildpack to use with the `cf push` command. If your Cloud Foundry deployment does not have the Node.js buildpack installed or the installed version is out of date, run `cf push APP-NAME -b https://github.com/cloudfoundry/nodejs-buildpack`, where `APP-NAME` is the name of your app, to push your app with the latest version of the buildpack.

For example:

```
$ cf push my-nodejs-app -b https://github.com/cloudfoundry/nodejs-buildpack
```

For more detailed information about deploying Node.js apps, see the following topics:

- Tips for Node.js Developers
- Environment Variables Defined by the Node Buildpack
- Configuring Service Connections for Node
- Node.js Buildpack Source Code on GitHub

Supported Versions

For a list of supported Node versions, see the Node.js Buildpack release notes on GitHub.

Specify a Node.js Version

To specify a Node.js version, set the `engines.node` in the `package.json` file to the semantic versioning specification (semver) range or the specific version of Node you are using.

Example showing a semver range:

```
"engines": {
  "node": "6.9.x"
}
```

Example showing a specific version:

```
"engines": {
  "node": "6.9.0"
}
```

If you try to use a version that is not currently supported, staging your app fails with the following error message:

```
Could not get translated url, exited with: DEPENDENCY_MISSING_IN_MANIFEST:...
!
!     exit
!
Staging failed: Buildpack compilation step failed
```

Specify an npm Version

To specify an npm version, set `engines.npm` in the `package.json` file to the semantic versioning specification (semver) range or the specific version of npm you are using:

Example showing a semver range:

```
"engines": {
  "node": "6.9.x",
  "npm": "2.15.x"
}
```

Example showing a specific version:

```
"engines": {
  "node": "6.9.0",
  "npm": "2.15.1"
}
```

If you do not specify an npm version, your app uses the default npm packaged with the Node.js version used by your app, as specified on the Node.js releases page.

If your environment cannot connect to the Internet and you specified a non-supported version of npm, the buildpack fails to download npm and you see the following error message:

```
We're unable to download the version of npm you've provided (...).
Please remove the npm version specification in package.json (...)
Staging failed: Buildpack compilation step failed
```

Vendor App Dependencies

To vendor dependencies for an app using the Node.js buildpack, run `npm install` from your app directory. This command vendors dependencies into the `node_modules` directory of your app directory.

For example, the following example vendors dependencies into the `my-nodejs-app/node_modules` directory:

```
$ cd my-nodejs-app
$ npm install
```

The `cf push` command uploads the vendored dependencies with the app.

Note: For an app to run in a disconnected environment, it must vendor its dependencies. For more information, see Disconnected environments.

Using Yarn in a Disconnected Environment

Versions 1.5.28 and later of the Node.js buildpack include the ability to use the package manager Yarn in a disconnected environment. To do so, you must mirror the Yarn registry locally:

```
$ cd APP-DIR
$ yarn config set yarn-offline-mirror ./npm-packages-offline-cache
$ cp ~/.yarnrc .
$ rm -rf node_modules/ yarn.lock # if they were previously generated
$ yarn install
```

When you push the app, the buildpack looks for an `npm-packages-offline-cache` directory at the top level of the app directory. If this directory exists, the buildpack runs Yarn in offline mode. Otherwise, it runs Yarn normally, which requires an Internet connection.

For more information about using an offline mirror with Yarn, see the Yarn Blog.

Integrity Check

By default, the Node.js buildpack uses npm to download dependencies. Note that npm does not perform integrity checks of the downloaded packages, which is a security risk.

If missing dependencies are detected, the buildpack runs `npm install` for non-vendored dependencies or `npm rebuild` for dependencies that are already vendored. This may result in code being downloaded and executed without verification.

You can use Yarn as an alternative that verifies dependencies.

OpenSSL Support

The nodejs-buildpack packages binaries of Node.js with OpenSSL that are statically linked. The Node.js buildpack supports Node.js 4.x and later, which relies on the Node.js release cycle to provide OpenSSL updates. The binary-builder enables static OpenSSL compilation.

Proxy Support

If you need to use a proxy to download dependencies during staging, you can set the `http_proxy` and/or `https_proxy` environment variables. For more information, see the Proxy Usage topic.

BOSH Configured Custom Trusted Certificate Support

Node.js hardcodes root CA certs in its source code. To use BOSH configured custom trusted certificates, a developer must pass the specified CAs to the tls.connect function as extra arguments.

Help and Support

Join the #buildpacks channel in our Slack community if you need any further assistance.

For more information about using and extending the Node.js buildpack in Cloud Foundry, see the Node.js GitHub repository.

You can find current information about this buildpack in the Node.js buildpack release page in GitHub.

Tips for Node.js Applications

Page last updated:

This topic provides Node-specific information to supplement the general guidelines in the Deploy an Application topic.

About the Node.js Buildpack

For information about using and extending the Node.js buildpack in Cloud Foundry, see the nodejs-buildpack repository.

You can find current information about this buildpack on the Node.js buildpack release page in GitHub.

The buildpack uses a default Node.js version. To specify the versions of Node.js and npm an app requires, edit the app's `package.json`, as described in "node.js and npm versions" in the nodejs-buildpack repository.

Application Package File

Cloud Foundry expects a `package.json` in your Node.js app. You can specify the version of Node.js you want to use in the `engine` node of your `package.json` file.

In general, Cloud Foundry supports the two most recent versions of Node.js. See the GitHub Node.js buildpack page for current information.

Example `package.json` file:

```
{
  "name": "first",
  "version": "0.0.1",
  "author": "Demo",
  "dependencies": {
    "express": "3.4.8",
    "consolidate": "0.10.0",
    "swig": "1.3.2"
  },
  "engines": {
    "node": "0.12.7",
    "npm": "2.7.4"
  }
}
```

Application Port

You must use the PORT environment variable to determine which port your app should listen on. To also run your app locally, set the default port as `3000`.

```
app.listen(process.env.PORT || 3000);
```

Low Memory Environments

When running node apps, you might notice that instances are occasionally restarted due to memory constraints. Node does not know how much memory it is allowed to use, and thus sometimes allows the garbage collector to wait past the allowed amount of memory. To resolve this issue, set the `OPTIMIZE_MEMORY` environment variable to `true`. This sets `max_old_space_size` based on the available memory in the instance.

```
$ cf set-env my-app OPTIMIZE_MEMORY true
```

Application Start Command

Node.js apps require a start command. You can specify the web start command for a Node.js app in a Procfile or in the app deployment manifest. For more information about Procfiles, see the Configuring a Production Server topic.

The first time you deploy, you are asked if you want to save your configuration. This saves a `manifest.yml` in your app with the settings you entered during the initial push. Edit the `manifest.yml` file and create a start command as follows:

```
---
applications:
- name: my-app
  command: node my-app.js
... the rest of your settings ...
```

Alternately, specify the start command with `cf push -c`.

```
$ cf push my-app -c "node my-app.js"
```

Application Bundling

You do not need to run `npm install` before deploying your app. Cloud Foundry runs it for you when your app is pushed. You can, if you prefer, run `npm install` and create a `node_modules` folder inside of your app.

Solve Discovery Problems

If Cloud Foundry does not automatically detect that your app is a Node.js app, you can override auto-detection by specifying the Node.js buildpack.

Add the buildpack into your `manifest.yml` and re-run `cf push` with your manifest:

```
---
applications:
- name: my-app
  buildpack: https://github.com/cloudfoundry/nodejs-buildpack
... the rest of your settings ...
```

Alternately, specify the buildpack on the command line with `cf push -b`:

```
$ cf push my-app -b https://github.com/cloudfoundry/nodejs-buildpack
```

Bind Services

Refer to Configure Service Connections for Node.js.

Environment Variables

You can access environments variable programmatically.

For example, you can obtain `VCAP_SERVICES` as follows:

```
process.env.VCAP_SERVICES
```

Environment variables available to you include both those defined by the system and those defined by the Node.js buildpack, as described below.

BUILD_DIR

Directory into which Node.js is copied each time a Node.js app is run.

CACHE_DIR

Directory that Node.js uses for caching.

PATH

The system path used by Node.js.

```
PATH=/home/vcap/app/bin:/home/vcap/app/node_modules/.bin:/bin:/usr/bin
```

Environment Variables Defined by the Node Buildpack

Page last updated:

Pivotal Application Service provides configuration information to applications through environment variables. This topic describes the additional environment variables provided by the Node buildpack.

For more information about the standard environment variables provided by Pivotal Application Service, see the Cloud Foundry Environment Variables topic.

Node Buildpack Environment Variables

The following table describes the environment variables provided by the Node buildpack.

Environment Variable	Description
`BUILD_DIR`	The directory where Node.js is copied each time a Node.js application runs.
`CACHE_DIR`	The directory Node.js uses for caching.
`PATH`	The system path used by Node.js: `PATH=/home/vcap/app/bin:/home/vcap/app/node_modules/.bin:/bin:/usr/bin`

Configuring Service Connections for Node.js

Page last updated:

This guide is for developers who wish to bind a data source to a Node.js application deployed and running on Cloud Foundry.

Parse VCAP_SERVICES for Credentials

You must parse the VCAP_SERVICES environment variable in your code to get the required connection details such as host address, port, user name, and password.

For example, if you are using PostgreSQL, your VCAP_SERVICES environment variable might look something like this:

```
{
  "mypostgres": [{
    "name": "myinstance",
    "credentials": {
      "uri": "postgres://myusername:mypassword@host.example.com:5432/serviceinstance"
    }
  }]
}
```

This example JSON is simplified; yours may contain additional properties.

Parse with cfenv

The `cfenv` package provides access to Cloud Foundry application environment settings by parsing all the relevant environment. The settings are returned as JavaScript objects. `cfenv` provides reasonable defaults when running locally, as well as when running as a Cloud Foundry application.

- https://www.npmjs.org/package/cfenv

Manual Parsing

First, parse the VCAP_SERVICES environment variable.

For example:

```
var vcap_services = JSON.parse(process.env.VCAP_SERVICES)
```

Then pull out the credential information required to connect to your service. Each service packages requires different information. If you are working with Postgres, for example, you will need a `uri` to connect. You can assign the value of the `uri` to a variable as follows:

```
var uri = vcap_services.mypostgres[0].credentials.uri
```

Once assigned, you can use your credentials as you would normally in your program to connect to your database.

Connecting to a Service

You must include the appropriate package for the type of services your application uses. For example:

- Rabbit MQ via the amqp module
- Mongo via the mongodb and mongoose modules
- MySQL via the mysql module
- Postgres via the pg module
- Redis via the redis module

Add the Dependency to package.json

Edit `package.json` and add the intended module to the `dependencies` section. Normally, only one would be necessary, but for the sake of the example we will add all of them:

```
{
  "name": "hello-node",
  "version": "0.0.1",
  "dependencies": {
    "express": "*",
    "mongodb": "*",
    "mongoose": "*",
    "mysql": "*",
    "pg": "*",
    "redis": "*",
    "amqp": "*"
  },
  "engines": {
    "node": "0.8.x"
  }
}
```

You must run `npm shrinkwrap` to regenerate your `npm-shrinkwrap.json` file after you edit `package.json`.

PHP Buildpack

Page last updated:

Use the PHP buildpack with PHP or HHVM runtimes.

Supported Software and Versions

The release notes page has a list of currently supported modules and packages.

- **PHP Runtimes**
  - php-cli
  - php-cgi
  - php-fpm
- **Third-Party Modules**
  - New Relic, in connected environments only.

Push an App

30 Second Tutorial

Getting started with this buildpack is easy. With the `cf` command line utility installed, open a shell, change directories to the root of your PHP files and push your application using the argument `-b https://github.com/cloudfoundry/php-buildpack.git`.

Example:

```
$ mkdir my-php-app
$ cd my-php-app
$ cat << EOF > index.php
<?php
  phpinfo();
?>
EOF
$ cf push -m 128M -b https://github.com/cloudfoundry/php-buildpack.git my-php-app
```

Change **my-php-app** in the above example to a unique name on your target Cloud Foundry instance to prevent a hostname conflict error and failed push.

The example above creates and pushes a test application, **my-php-app**, to Cloud Foundry. The `-b` argument instructs CF to use this buildpack. The remainder of the options and arguments are not specific to the buildpack, for questions on those consult the output of `cf help push`.

Here's a breakdown of what happens when you run the example above.

- On your PC:
  - It will create a new directory and one PHP file, which just invokes `phpinfo()`
  - Run `cf` to push your application. This will create a new application with a memory limit of 128M (more than enough here) and upload our test file.
- Within Cloud Foundry:
  - The buildpack is executed.
  - Application files are copied to the `htdocs` folder.
  - Apache HTTPD & PHP are downloaded, configured with the buildpack defaults and run.
  - Your application is accessible at the URL `http://my-php-app.example.com` (Replacing `example.com` with the domain of your public CF provider or private instance).

More information about deploying

While the *30 Second Tutorial* shows how quick and easy it is to get started using the buildpack, it skips over quite a bit of what you can do to adjust, configure and extend the buildpack. The following sections and links provide a more in-depth look at the buildpack.

Features

Here are some special features of the buildpack.

- Supports running commands or migration scripts prior to application startup.
- Supports an extension mechanism that allows the buildpack to provide additional functionality.
- Allows for application developers to provide custom extensions.
- Easy troubleshooting with the `BP_DEBUG` environment variable.
- Download location is configurable, allowing users to host binaries on the same network (i.e. run without an Internet connection)
- Smart session storage, defaults to file w/sticky sessions but can also use redis for storage.

Examples

Here are some example applications that can be used with this buildpack.

- php-info This app has a basic index page and shows the output of `phpinfo()`.
- PHPMyAdmin A deployment of PHPMyAdmin that uses bound MySQL services.
- PHPPgAdmin A deployment of PHPPgAdmin that uses bound PostgreSQL services.
- Drupal A deployment of Drupal that uses bound MySQL service.
- CodeIgniter CodeIgniter tutorial application running on CF.
- Stand Alone An example which runs a standalone PHP script.
- pgbouncer An example which runs the PgBouncer process in the container to pool database connections.
- phalcon An example which runs a Phalcon based application.
- composer An example which uses Composer.

Advanced Topics

See the following topics:

- Tips for PHP Developers
- Getting Started Deploying PHP Apps
- PHP Buildpack Configuration
- Composer
- Sessions
- New Relic

You can find the source for the buildpack on GitHub: https://github.com/cloudfoundry/php-buildpack

Proxy Support

If you need to use a proxy to download dependencies during staging, you can set the `http_proxy` and/or `https_proxy` environment variables. For more information, see the Proxy Usage Docs.

BOSH Configured Custom Trusted Certificate Support

For versions of PHP 5.6.0 and later, the default certificate location is `/usr/lib/ssl/certs`, which symlinks to `/etc/ssl/certs`. Your platform operator can configure the platform to add the custom certificates into the application container.

Help and Support

Join the #buildpacks channel in our Slack community if you need any further assistance.

For more information about using and extending the PHP buildpack in Cloud Foundry, see the php-buildpack GitHub repository.

You can find current information about this buildpack on the PHP buildpack release page in GitHub.

License

The Cloud Foundry PHP Buildpack is released under version 2.0 of the Apache License.

Tips for PHP Developers

Page last updated:

About the PHP Buildpack

For information about using and extending the PHP buildpack in Cloud Foundry, see the php-buildpack Github repository.

You can find current information about this buildpack on the PHP buildpack release page in GitHub.

The buildpack uses a default PHP version specified in `.defaults/options.json` under the `PHP_VERSION` key.

To change the default version, specify the `PHP_VERSION` key in your app's `.bp-config/options.json` file.

Getting Started Deploying PHP Apps

Page last updated:

This document is intended to guide you through the process of deploying PHP applications to Pivotal Application Service. If you experience a problem with deploying PHP apps, check the Troubleshooting section below.

Getting Started

Prerequisites

- Basic PHP knowledge
- The Cloud Foundry Command Line Interface (cf CLI) installed on your workstation

A First PHP Application

```
$ mkdir my-php-app
$cd my-php-app
$ cat << EOF > index.php
<?php
  phpinfo();
?>
EOF
$ cf push my-php-app -m 128M
```

Change “my-php-app” to a unique name, otherwise you get an error and the push fails.

The example above creates and pushes a test application, “my-php-app”, to Cloud Foundry.

Here is a breakdown of what happens when you run the example above:

- On your workstation…
  - It creates a new directory and one PHP file, which calls `phpinfo()`
  - Run `cf push` to push your application. This will create a new application with a memory limit of 128M and upload our test file.
- On Cloud Foundry…
  - The buildpack detects that your app is a php app
  - The buildpack is executed.
    - Application files are copied to the `htdocs` folder.
  - Apache HTTPD & PHP are downloaded, configured with the buildpack defaults, and run.
  - Your application is accessible at the default route. Use `cf app my-php-app` to view the url of your new app.

Folder Structure

The easiest way to use the buildpack is to put your assets and PHP files into a directory and push it to PAS. This way, the buildpack will take your files and automatically move them into the `WEBDIR` (defaults to `htdocs`) folder, which is the directory where your chosen web server looks for the files.

URL Rewriting

If you select Apache as your web server, you can include `.htaccess` files with your application.

Alternatively, you can provide your own Apache or Nginx configurations.

Prevent Access To PHP Files

The buildpack will put all of your files into a publicly accessible directory. In some cases, you might want to have PHP files that are not publicly accessible but are on the include_path. To do that, create a `lib` directory in your project folder and place your protected files there.

For example:

```
$ ls -lRh
total 0
-rw-r--r--  1 daniel  staff    0B Feb 27 21:40 images
-rw-r--r--  1 daniel  staff    0B Feb 27 21:39 index.php
drwxr-xr-x  3 daniel  staff  102B Feb 27 21:40 lib

./lib:
total 0
-rw-r--r--  1 daniel  staff    0B Feb 27 21:40 my.class.php  <-- not public, http://app.cfapps.io/lib/my.class.php == 404
```

This comes with a catch. If your project legitimately has a `lib` directory, these files will not be publicly available because the buildpack does not copy a top-level `lib` directory into the `WEBDIR` folder. If your project has a `lib` directory that needs to be publicly available, then you have two options as follows:

Option #1

In your project folder, create an `htdocs` folder (or whatever you've set for `WEBDIR`). Then move any files that should be publicly accessible into this directory. In the example below, the `lib/controller.php` file is publicly accessible.

Example:

```
$ ls -lRh
total 0
drwxr-xr-x  7 daniel  staff  238B Feb 27 21:48 htdocs

./htdocs:  <--  create the htdocs directory and put your files there
total 0
-rw-r--r--  1 daniel  staff    0B Feb 27 21:40 images
-rw-r--r--  1 daniel  staff    0B Feb 27 21:39 index.php
drwxr-xr-x  3 daniel  staff  102B Feb 27 21:48 lib

./htdocs/lib:   <--  anything under htdocs is public, including a lib directory
total 0
-rw-r--r--  1 daniel  staff    0B Feb 27 21:48 controller.php
```

Given this setup, it is possible to have both a public `lib` directory and a protected `lib` directory. The following example demonstrates this setup:

Example:

```
$ ls -lRh
total 0
drwxr-xr-x  7 daniel  staff  238B Feb 27 21:48 htdocs
drwxr-xr-x  3 daniel  staff  102B Feb 27 21:51 lib

./htdocs:
total 0
-rw-r--r--  1 daniel  staff    0B Feb 27 21:40 images
-rw-r--r--  1 daniel  staff    0B Feb 27 21:39 index.php
drwxr-xr-x  3 daniel  staff  102B Feb 27 21:48 lib

./htdocs/lib:  <-- public lib directory
total 0
-rw-r--r--  1 daniel  staff    0B Feb 27 21:48 controller.php

./lib: <-- protected lib directory
total 0
-rw-r--r--  1 daniel  staff    0B Feb 27 21:51 my.class.php
```

Option #2

The second option is to pick a different name for the `LIBDIR`. This is a configuration option that you can set (it defaults to `lib`). Thus if you set it to something else such as `include`, your application's `lib` directory would no longer be treated as a special directory and it would be placed into `WEBDIR` (i.e. become public).

Other Folders

Beyond the `WEBDIR` and `LIBDIR` directories, the buildpack also supports a `.bp-config` directory and a `.extensions` directory.

The `.bp-config` directory should exist at the root of your project directory and it is the location of application-specific configuration files. Application-specific configuration files override the default settings used by the buildpack. This link explains application configuration files in depth.

The `.extensions` directory should also exist at the root of your project directory and it is the location of application-specific custom extensions. Application-specific custom extensions allow you, the developer, to override or enhance the behavior of the buildpack. This link explains extensions in more detail.

Troubleshooting

There are a couple of easy ways to debug the buildpack:

1. Check the output from the buildpack. It writes some basic information to stdout, like the files that are being downloaded. It writes information should something fail, specifically, stack traces.

2. Check the logs from the buildpack. The buildpack writes logs to disk. Retrieve them with the command, as the following example shows:

```
$ cf ssh APP
$ cat app/.bp/logs/bp.log
```

This log is more detailed than the stdout output, but is still terse.

Set the `BP_DEBUG` environment variable to `true` for more verbose logging. This instructs the buildpack to set its log level to `DEBUG` and it writes to stdout. Follow Environment Variables documentation to set `BP_DEBUG`.

PHP Buildpack Configuration

Page last updated:

Defaults

The PHP buildpack stores all of its default configuration settings in the defaults directory.

options.json

The `options.json` file is the configuration file for the buildpack itself. It instructs the buildpack what to download, where to download it from, and how to install it. It allows you to configure package names and versions (i.e. PHP, HTTPD, or Nginx versions), the web server to use (HTTPD, Nginx, or None), and the PHP extensions that are enabled.

The buildpack overrides the default `options.json` file with any configuration it finds in the `.bp-config/options.json` file of your application.

Below is an explanation of the common options you might need to change.

Variable	Explanation
WEB_SERVER	Sets the web server to use. Must be one of `httpd`, `nginx`, or `none`. This value defaults to `httpd`.
HTTPD_VERSION	Sets the version of Apache HTTPD to use. Currently the build pack supports the latest stable version. This value will default to the latest release that is supported by the build pack.
ADMIN_EMAIL	The value used in HTTPD's configuration for ServerAdmin
NGINX_VERSION	Sets the version of Nginx to use. By default, the buildpack uses the latest stable version.
PHP_VERSION	Sets the version of PHP to use. Set to a minor instead of a patch version, such as `"{PHP_70_LATEST}"`. See options.json.
PHP_EXTENSIONS	(DEPRECATED) A list of the extensions to enable. `bz2`, `zlib`, `curl`, and `mcrypt` are enabled by default.
PHP_MODULES	A list of the modules to enable. No modules are explicitly enabled by default, however the buildpack automatically chooses `fpm` or `cli`. You can explicitly enable any or all of: `fpm`, `cli`, `cgi`, and `pear`.
ZEND_EXTENSIONS	A list of the Zend extensions to enable. Nothing is enabled by default.
APP_START_CMD	When the `WEB_SERVER` option is set to 'none,' this command is used to start your app. If `WEB_SERVER` and `APP_START_CMD` are not set, then the buildpack searches for `app.php`, `main.php`, `run.php`, or `start.php` (in that order). This option accepts arguments.
WEBDIR	The root directory of the files served by the web server specified in `WEB_SERVER`. Defaults to `htdocs`. Other common settings are `public`, `static`, or `html`. Path is relative to the root of your application.
LIBDIR	This path is added to PHP's `include_path`. Defaults to `lib`. Path is relative to the root of your application.
HTTP_PROXY	The buildpack downloads uncached dependencies using HTTP. If you are using a proxy for HTTP access, set its URL here.
HTTPS_PROXY	The buildpack downloads uncached dependencies using HTTPS. If you are using a proxy for HTTPS access, set its URL here.
ADDITIONAL_PREPROCESS_CMDS	A list of additional commands that will run prior to the application starting. For example, you might use this command to run migration scripts or static caching tools before the application launches.

For details about supported versions, please read the release notes for your buildpack version.

HTTPD, Nginx, and PHP configuration

The buildpack automatically configures HTTPD, Nginx, and PHP for your application. This section explains how to modify the configuration.

The `.bp-config` directory in your application can contain configuration overrides for these components. Name the directories `httpd`, `nginx`, and `php`. We recommend that you use php.ini.d or fpm.d).

NOTE: If you override the php.ini or php-fpm.ini files many other forms of configuration will not work.

For example: `.bp-config httpd nginx php`

Each directory can contain configuration files that the component understands.

For example, to change HTTPD logging configuration:

```
$ ls -l .bp-config/httpd/extra/
total 8
-rw-r--r--  1 daniel  staff  396 Jan  3 08:31 httpd-logging.conf
```

In this example, the `httpd-logging.conf` file overrides the one provided by the buildpack. We recommend that you copy the default from the buildpack and modify it.

The default configuration files are found in the PHP Buildpack `/defaults/config` directory

Take care when modifying configurations, as it might cause your application to fail, or cause Cloud Foundry to fail to stage your application.

You can add your own configuration files. The components will not know about these, so you must ensure that they are included. For example, you can add an include directive to the httpd configuration to include your file:

```
ServerRoot "${HOME}/httpd"
Listen ${PORT}
ServerAdmin "${HTTPD_SERVER_ADMIN}"
ServerName "0.0.0.0"
DocumentRoot "${HOME}/#{WEBDIR}"
Include conf/extra/httpd-modules.conf
Include conf/extra/httpd-directories.conf
Include conf/extra/httpd-mime.conf
Include conf/extra/httpd-logging.conf
Include conf/extra/httpd-mpm.conf
Include conf/extra/httpd-default.conf
Include conf/extra/httpd-remoteip.conf
Include conf/extra/httpd-php.conf
Include conf/extra/httpd-my-special-config.conf # This line includes your additional file.
```

.bp-config/php/php.ini.d/

The buildpack will add any `.bp-config/php/php.ini.d/<file>.ini` files it finds in the application to the PHP configuration. This can be used to change any value which is acceptable to php.ini (list of directives).

For example adding a file `.bp-config/php/php.ini.d/something.ini` to your app, with contents

```
default_charset="UTF-8"
default_mimetype="text/xhtml"
```

would override both the default charset and mimetype.

Precedence

In order of highest precedence, php config values come from.

- php scripts using ini_set() to manually override config files
- user.ini files (for local values)
- .bp-config/php/php.ini.d overrides master value but, local value is still the one from user.ini

.bp-config/php/fpm.d/

The buildpack will add any `.bp-config/php/fpm.d` files it finds in the application to the PHP-FPM configuration. This can bes used to change any value which is acceptable to `php-fpm.ini` (list of directives)

PHP Extensions

The buildpack will add any `.bp-config/php/php.ini.d/<file>.ini` files it finds in the application to the PHP configuration. This can be used to enable PHP or ZEND extensions. For example,

```
extension=redis.so
extension=gd.so
zend_extension=opcache.so
```

If an extension is already present and enabled in the compiled php, for example `intl`, you do not need to explicitly enable it to use that extension.

PHP_EXTENSIONS vs. ZEND_EXTENSIONS

PHP has two kinds of extensions, "PHP extensions" and "Zend extensions". These hook into the PHP executable in different ways.

For this reason, they are specified differently in ini files. Your app will fail if a Zend Extension is specified as a PHP Extension, or vice versa.

If you see this error:

```
php-fpm | [warn-ioncube] The example Loader is a Zend-Engine extension and not a module (pid 40)
php-fpm | [warn-ioncube] Please specify the Loader using 'zend_extension' in php.ini (pid 40)
php-fpm | NOTICE: PHP message: PHP Fatal error:  Unable to start example Loader module in Unknown on line 0
```

Try moving the `example` extension from `extension` to `zend_extension`, then re-pushing your application.

If you see this error:

```
NOTICE: PHP message: PHP Warning:  example MUST be loaded as a Zend extension in Unknown on line 0
```

Try moving the `example` extension from `zend_extension` to `extension`, then re-pushing your application.

PHP Modules

The following modules can be included by adding it to the `PHP_MODULES` list:

- `cli`, installs `php` and `phar`
- `fpm`, installs `PHP-FPM`
- `cgi`, installs `php-cgi`
- `pear`, installs Pear

By default, the buildpack installs the `cli` module when you push a standalone application, and it installs the `fpm` module when you run a web application. You must specify `cgi` and `pear` if you want them installed.

Buildpack Extensions

The buildpack comes with extensions for its default behavior. These are the HTTPD, Nginx, PHP, and NewRelic extensions.

The buildpack is designed with an extension mechanism, allowing application developers to add behavior to the buildpack without modifying the buildpack code.

When an application is pushed, the buildpack runs any extensions found in the `.extensions` directory of your application.

The Developer Documentation explains how to write extensions.

Composer

Page last updated:

Composer is activated when you supply a `composer.json` or `composer.lock` file. A `composer.lock` is not required, but is strongly recommended for consistent deployments.

You can require dependencies for packages and extensions. Extensions must be prefixed with the standard `ext-`. If you reference an extension that is available to the buildpack, it will automatically be installed. See the main README for a list of supported extensions.

The buildpack uses the version of PHP specified in your `composer.json` or `composer.lock` file. Composer settings override the version set in the `options.json` file.

The PHP buildpack supports a subset of the version formats supported by Composer. The buildpack supported formats are:

Example	Expected Version
5.3.*	latest 5.4.x release (5.3 is not supported)
>=5.3	latest 5.4.x release (5.3 is not supported)
5.4.*	latest 5.4.x release
>=5.4	latest 5.4.x release
5.5.*	latest 5.5.x release
>=5.5	latest 5.5.x release
5.4.x	specific 5.4.x release that is listed
5.5.x	specific 5.5.x release that is listed

Configuration

The buildpack runs with a set of default values for Composer. You can adjust these values by adding a `.bp-config/options.json` file to your application and setting any of the following values in it.

Variable	Explanation
COMPOSER_VERSION	The version of Composer to use. It defaults to the latest bundled with the buildpack.
COMPOSER_INSTALL_OPTIONS	A list of options that should be passed to `composer install`. This defaults to `--no-interaction --no-dev --no-progress`. The `--no-progress` option must be used due to the way the buildpack calls Composer.
COMPOSER_VENDOR_DIR	Allows you to override the default value used by the buildpack. This is passed through to Composer and instructs it where to create the `vendor` directory. Defaults to `{BUILD_DIR}/{LIBDIR}/vendor`.
COMPOSER_BIN_DIR	Allows you to override the default value used by the buildpack. This is passed through to Composer and instructs it where to place executables from packages. Defaults to `{BUILD_DIR}/php/bin`.
COMPOSER_CACHE_DIR	Allows you to override the default value used by the buildpack. This is passed through to Composer and instructs it where to place its cache files. Generally you should not change this value. The default is `{CACHE_DIR}/composer` which is a subdirectory of the cache folder passed in to the buildpack. Composer cache files will be restored on subsequent application pushes.

By default, the PHP buildpack uses the `composer.json` and `composer.lock` files that reside inside the root directory, or in the directory specified as `WEBDIR` in your `options.json`. If you have composer files inside your app, but not in the default directories, use a `COMPOSER_PATH` environment variable for your app to specify this custom location, relative to the app root directory. Note that the `composer.json` and `composer.lock` files must be in the same directory.

Github API Request Limits

Composer uses Github's API to retrieve zip files for installation into the application folder. If you do not vendor dependencies before pushing an app, Composer will fetch dependencies during staging using the Github API.

Github's API is request-limited. If you reach your daily allowance of API requests (typically 60), Github's API returns a `403` error and staging fails.

There are two ways to avoid the request limit:

1. Vendor dependencies before pushing your application.
2. Supply a Github OAuth API token.

Vendor Dependencies

To vendor dependencies, you must run `composer install` before you push your application. You might also need to configure `COMPOSER_VENDOR_DIR` to “vendor”.

Supply a Github Token

Composer can use Github API OAuth tokens, which increase your request limit, typically to 5000 per day.

During staging, the buildpack looks for this token in the environment variable `COMPOSER_GITHUB_OAUTH_TOKEN`. If you supply a valid token, Composer uses it. This mechanism does not work if the token is invalid.

To supply the token, you can use either of the following methods:

1. `cf set-env YOUR_APP_NAME COMPOSER_GITHUB_OAUTH_TOKEN "OAUTH_TOKEN_VALUE"`
2. Add the token to the `env` block of your application manifest.

Buildpack Staging Environment

Composer runs in the buildpack staging environment. Variables set with `cf set-env` or with a manifest.yml ‘env’ block are visible to Composer.

For example:

```
$ cf push a_symfony_app --no-start
$ cf set-env a_symfony_app SYMFONY_ENV "prod"
$ cf start a_symfony_app
```

In this example, `a_symfony_app` is supplied with an environment variable, `SYMFONY_ENV`, which is visible to Composer and any scripts started by Composer.

Non-configurable Environment Variables

User-assigned environment variables are applied to staging and runtime. Unfortunately, `LD_LIBRARY_PATH` and `PHPRC` must be different for staging and runtime. The buildpack takes care of setting these variables, which means user values for these variables are ignored.

Sessions

Page last updated:

Usage

When your application has one instance, it's mostly safe to use the default session storage, which is the local file system. You would only see problems if your single instance crashes as the local file system would go away and you'd lose your sessions. For many applications, this will work just fine but please consider how this will impact your application.

If you have multiple application instances or you need a more robust solution for your application, then you'll want to use Redis or Memcached as a backing store for your session data. The build pack supports both and when one is bound to your application it will detect it and automatically configure PHP to use it for session storage.

By default, there's no configuration necessary. Create a Redis or Memcached service, make sure the service name contains `redis-sessions` or `memcached-sessions` and then bind the service to the application.

Example:

```
$ cf create-service redis some-plan app-redis-sessions
$ cf bind-service app app-redis-sessions
$ cf restage app
```

If you want to use a specific service instance or change the search key, you can do that by setting either `REDIS_SESSION_STORE_SERVICE_NAME` or `MEMCACHED_SESSION_STORE_SERVICE_NAME` in `.bp-config/options.json` to the new search key. The session configuration extension will then search the bound services by name for the new session key.

Configuration Changes

When detected, the following changes will be made.

Redis

- the `redis` PHP extension will be installed, which provides the session save handler
- `session.name` will be set to `PHPSESSIONID` which disables sticky sessions
- `session.save_handler` is configured to `redis`
- `session.save_path` is configured based on the bound credentials, for example `tcp://host:port?auth=pass`

Memcached

- the `memcached` PHP extension will be installed, which provides the session save handler
- `session.name` will be set to `PHPSESSIONID` which disables sticky sessions
- `session.save_handler` is configured to `memcached`
- `session.save_path` is configured based on the bound credentials (i.e. `PERSISTENT=app_sessions host:port`)
- `memcached.sess_binary` is set to `On`
- `memcached.use_sasl` is set to `On`, which enables authentication
- `memcached.sess_sasl_username` and `memcached.sess_sasl_password` are set with the service credentials

New Relic

Page last updated:

New Relic collects analytics about application and client-side performance.

Configuration

You can configure New Relic for the PHP buildpack in one of two ways:

- With a license key
- With a Cloud Foundry service

With a License Key

Use this method if you already have a New Relic account,

1. In a web browser. navigate to the New Relic website to find your license key.
2. Set the value of the environment variable `NEWRELIC_LICENSE` to your New Relic license key, either through the manifest.yml file or by running the `cf set-env` command.

For more information, see https://github.com/cloudfoundry/php-buildpack#supported-software

With a Cloud Foundry Service

To configure New Relic for the PHP buildpack with a Cloud Foundry service, bind a New Relic service to the app. The buildpack automatically detects and configures New Relic.

Your `VCAP_SERVICES` environment variable must contain a service named `newrelic`, the `newrelic` service must contain a key named `credentials`, and the `credentials` key must contain named `licenseKey`.

NOTE: You cannot configure New Relic for the PHP buildpack with user-provided services.

Python Buildpack

Page last updated:

Push an App

Cloud Foundry automatically uses this buildpack if it detects a `requirements.txt` or `setup.py` file in the root directory of your project.

If your Cloud Foundry deployment does not have the Python Buildpack installed, or the installed version is out of date, you can use the latest version by specifying it with the `-b` option when you push your app. For example:

```
$ cf push my_app -b https://github.com/cloudfoundry/buildpack-python.git
```

Supported Versions

You can find the list of supported Python versions in the Python buildpack release notes.

Specify a Python Version

You can specify a version of the Python runtime by including it within a `runtime.txt` file. For example:

```
$ cat runtime.txt
python-3.5.2
```

The buildpack only supports the stable Python versions, which are listed in the `manifest.yml` and Python buildpack release notes.

If you try to use a binary that is not currently supported, staging your app fails and you see the following error message:

```
Could not get translated url, exited with: DEPENDENCY_MISSING_IN_MANIFEST: ...
 !
 !     exit
 !
Staging failed: Buildpack compilation step failed
```

Specify a Start Command

The Python buildpack does not generate a default start command for your applications.

To stage with the Python buildpack and start an application, do one of the following:

- Supply a Procfile. For more information about Procfiles, see the Configuring a Production Server topic. The following example Procfile specifies `gunicorn` as the start command for a web app running on Gunicorn:

  ```
  web: gunicorn SERVER-NAME:APP-NAME
  ```

- Specify a start command with `-c`. The following example specifies `waitress-serve` as the start command for a web app running on Waitress:

  ```
  $ cf push python-app -c "waitress-serve --port=$PORT DJANGO-WEB-APP.wsgi:MY-APP"
  ```

- Specify a start command in the application manifest by setting the `command` attribute. For more information, see the Deploying with Application Manifests topic.

Vendor App Dependencies

If you are deploying in an environment that is disconnected from the Internet, your application must vendor its dependencies.

For the Python buildpack, use `pip`:

```
$ cd YOUR-APP-DIR
$ mkdir -p vendor

# vendors all the pip *.tar.gz into vendor/
$ pip install --download vendor -r requirements.txt --no-binary :all:
```

`cf push` uploads your vendored dependencies. The buildpack installs them directly from the `vendor/` directory.

Note: To ensure proper installation of dependencies, we recommend non-binary vendored dependencies. The above `pip install` command achieves this.

Parse Environment Variables

The `cfenv` package provides access to Cloud Foundry application environment settings by parsing all the relevant environment variables. The settings are returned as a class instance. See https://github.com/jmcarp/py-cfenv for more information.

Miniconda Support (starting in buildpack version 1.5.6)

To use miniconda instead of pip for installing dependencies, place an `environment.yml` file in the root directory.

For examples, see our sample apps:

- Using Python 2 with miniconda
- Using Python 3 with miniconda

Pipenv Support (starting in buildpack version 1.5.19)

To use Pipenv instead of pip (directly) for installing dependencies, place a `Pipfile` in the root directory. Easiest to let Pipenv generate this for you.

Proxy Support

If you need to use a proxy to download dependencies during staging, you can set the `http_proxy` and `https_proxy` environment variables. For more information, see the Proxy Usage Documentation.

BOSH Configured Custom Trusted Certificate Support

Versions of Python 2.7.9 and later use certificates stored in `/etc/ssl/certs`. Your platform operator can configure the platform to add the custom certificates into the application container.

Help and Support

Join the #buildpacks channel in our Slack community if you need any further assistance.

For more information about using and extending the Python buildpack in Cloud Foundry, see the python-buildpack GitHub repository.

You can find current information about this buildpack on the Python buildpack release page in GitHub.

Ruby Buildpack

Page last updated:

Push Apps

Cloud Foundry uses the Ruby buildpack if your app has a `Gemfile` and `Gemfile.lock` in the root directory. The Ruby buildpack uses Bundler to install your dependencies.

If your Cloud Foundry deployment does not have the Ruby buildpack installed or the installed version is out of date, push your app with the `-b` option to specify the buildpack:

```
$ cf push MY-APP -b https://github.com/cloudfoundry/ruby-buildpack.git
```

For more detailed information about deploying Ruby applications see the following topics:

- Tips for Ruby Developers
- Getting Started Deploying Apps
- Configuring Rake Tasks for Deployed Apps
- Environment Variables Defined by the Ruby Buildpack
- Configuring Service Connections for Ruby
- Support for Windows Gemfiles

You can find the source for the Ruby buildpack in the Ruby buildpack repository on GitHub.

Supported Versions

You can find supported Ruby versions in the release notes for the Ruby buildpack on GitHub.

Specify a Ruby Version

Specify specific versions of the Ruby runtime in the `Gemfile` for your app as described in the sections below.

MRI

For MRI, specify the version of Ruby in your `Gemfile` as follows:

```
ruby '~> 2.2.3'
```

With this example declaration in the `Gemfile`, if Ruby versions `2.2.4`, `2.2.5`, and `2.3.0` are present in the Ruby buildpack, the app uses Ruby version `2.2.5`.

For more information about the `ruby` directive for Bundler Gemfiles, see the Bundler documentation.

Note: If you use v1.6.18 or earlier, you must specify an exact version, such as `ruby '2.2.3'`. In Ruby Buildpack v1.6.18 and earlier, Rubygems do not support version operators for the `ruby` directive.

JRuby

For JRuby, specify the version of Ruby in your `Gemfile` based on the version of JRuby your app uses.

JRuby version `1.7.x` supports either `1.9` mode or `2.0` mode.

- For `1.9` mode, use the following: `ruby ruby '1.9.3', :engine => 'jruby', :engine_version => '1.7.25'`
- For `2.0` mode, use the following: `ruby ruby '2.0.0', :engine => 'jruby', :engine_version => '1.7.25'`

For Jruby version `>= 9.0`, use the following: `ruby ruby '2.2.3', :engine => 'jruby', :engine_version => '9.0.5.0'`

The Ruby buildpack only supports the stable Ruby versions listed in the `manifest.yml` and release notes for the Ruby buildpack on GitHub.

If you try to use a binary that is not supported, staging your app fails with the following error message:

```
Could not get translated url, exited with: DEPENDENCY_MISSING_IN_MANIFEST: ...
!
!    exit
!
Staging failed: Buildpack compilation step failed
```

Note: The Ruby buildpack does not support he pessimistic version operator `~>` on the Gemfile `ruby` directive for JRuby.

Vendor App Dependencies

As stated in the Disconnected Environments documentation, your application must 'vendor' its dependencies.

For the Ruby buildpack, use the `bundle package --all` command in Bundler to vendor the dependencies.

Example:

```
$ cd my-app-directory
bundle package --all
```

The `cf push` command uploads your vendored dependencies. The Ruby buildpack compiles any dependencies requiring compilation while staging your app.

Buildpack Logging and Application Logging

The Ruby buildpack only runs during the staging process, and only logs what is important to staging, such as what is being downloaded, what the configuration is, and work that the buildpack does on your application.

The buildpack stops logging when the staging process finishes. The Loggregator handles application logging.

Your application must write to STDOUT or STDERR for its logs to be included in the Loggregator stream. For more information, see the Application Logging in Cloud Foundry topic.

If you are deploying a Rails application, the buildpack may or may not automatically install the necessary plugin or gem for logging, depending on the Rails version of the application:

- Rails 2.x: The buildpack automatically installs the `rails_log_stdout` plugin into the application. For more information about the `rails_log_stdout` plugin, refer to the Github README.
- Rails 3.x: The buildpack automatically installs the `rails_12factor` gem if it is not present and issues a warning message. You must add the `rails_12factor` gem to your `Gemfile` to quiet the warning message. For more information about the `rails_12factor` gem, refer to the Github README.
- Rails 4.x: The buildpack only issues a warning message that the `rails_12factor` gem is not present, but does not install the gem. You must add the `rails_12factor` gem to your `Gemfile` to quiet the warning message. For more information about the `rails_12factor` gem, refer to the Github README.

For more information about the `rails_12factor` gem, refer to the Github README.

Proxy Support

If you need to use a proxy to download dependencies during staging, you can set the `http_proxy` and/or `https_proxy` environment variables. For more information, see the Proxy Usage Docs.

BOSH Configured Custom Trusted Certificate Support

Ruby uses certificates stored in `/etc/ssl/certs`. Your platform operator can configure the platform to add the custom certificates into the app container.

Help and Support

Join the #buildpacks channel in our Slack community if you need any further assistance.

For more information about using and extending the Ruby buildpack in Cloud Foundry, see the Ruby buildpack repository on GitHub.

You can find current information about the Ruby buildpack in the release notes for the Ruby buildpack on GitHub.

Tips for Ruby Developers

Page last updated:

This page has information specific to deploying Rack, Rails, or Sinatra apps.

App Bundling

You must run Bundler to create a `Gemfile` and a `Gemfile.lock`. These files must be in your app before you push to Cloud Foundry.

Rack Config File

For Rack and Sinatra, you must have a `config.ru` file. For example:

```
require './hello_world'
run HelloWorld.new
```

Asset Precompilation

Cloud Foundry supports the Rails asset pipeline. If you do not precompile assets before deploying your app, Cloud Foundry precompiles them when staging the app. Precompiling before deploying reduces the time it takes to stage an app.

Use the following command to precompile assets before deployment:

```
$ rake assets:precompile
```

Note that the Rake precompile task reinitializes the Rails app. This could pose a problem if initialization requires service connections or environment checks that are unavailable during staging. To prevent reinitialization during precompilation, add the following line to `application.rb`:

```
config.assets.initialize_on_precompile = false
```

If the `assets:precompile` task fails, Cloud Foundry uses live compilation mode, the alternative to asset precompilation. In this mode, assets are compiled when they are loaded for the first time. You can force live compilation by adding the following line to `application.rb`.

```
Rails.application.config.assets.compile = true
```

Running Rake Tasks

Cloud Foundry does not provide a mechanism for running a Rake task on a deployed app. If you need to run a Rake task that must be performed in the Cloud Foundry environment, rather than locally before deploying or redeploying, you can configure the command that Cloud Foundry uses to start the app to invoke the Rake task.

An app start command is configured in the app manifest file, `manifest.yml`, using the `command` attribute.

For more information about app manifests and supported attributes, see the Deploying with Application Manifests topic.

Example: Invoking a Rake database migration task at app startup

The following is an example of the “migrate frequently” method described in the Migrating a Database in Cloud Foundry topic.

1. If a Rakefile does not exist, create one and add it to your app directory.

2. In your Rakefile, add a Rake task to limit an idempotent command to the first instance of a deployed app:

```
namespace :cf do
  desc "Only run on the first application instance"
  task :on_first_instance do
    instance_index = JSON.parse(ENV["VCAP_APPLICATION"])["instance_index"] rescue nil
    exit(0) unless instance_index == 0
  end
end
```

3. Add the task to the `manifest.yml` file, referencing the idempotent command `rake db:migrate` with the `command` attribute.

```
---
applications:
- name: my-rails-app
  command: bundle exec rake cf:on_first_instance db:migrate && bundle exec rails s -p $PORT -e $RAILS_ENV
```

4. Update the app using `cf push`.

Rails 3 Worker Tasks

This section shows you how to create and deploy an example Rails app that uses a worker library to defer a task that a separate app executes.

The guide also describes how to scale the resources available to the worker app.

> **Note**: Most worker tasks do not serve external requests. Use the `--no-route` flag with the `cf push` command, or `no-route: true` in the app manifest, to suppress route creation and remove existing routes.

Choose a Worker Task Library

You must choose a worker task library. The table below summarizes the three main libraries available for Ruby / Rails:

Library	Description
Delayed::Job	A direct extraction from Shopify where the job table is responsible for a multitude of core tasks.
Resque	A Redis-backed library for creating background jobs, placing those jobs on multiple queues, and processing them later.
Sidekiq	Uses threads to handle many messages at the same time in the same process. It does not require Rails, but integrates tightly with Rails 3 to simplify background message processing. This library is Redis-backed and semi-compatible with Resque messaging.

For other alternatives, see https://www.ruby-toolbox.com/categories/Background_Jobs.

Create an Example App

For the purposes of the example app, we use Sidekiq.

First, create a Rails app with an arbitrary model named “Things”:

```
$ rails create rails-sidekiq
$ cd rails-sidekiq
$ rails g model Thing title:string description:string
```

Add `sidekiq` and `uuidtools` to the Gemfile:

```
source 'https://rubygems.org'

gem 'rails', '3.2.9'
gem 'mysql2'

group :assets do
  gem 'sass-rails',   '~> 3.2.3'
  gem 'coffee-rails', '~> 3.2.1'
  gem 'uglifier', '>= 1.0.3'
end

gem 'jquery-rails'
gem 'sidekiq'
gem 'uuidtools'
```

Install the bundle.

```
$ bundle install
```

In `app/workers`, create a worker for Sidekiq to carry out its tasks:

```
$ touch app/workers/thing_worker.rb
```

```
class ThingWorker

  include Sidekiq::Worker

  def perform(count)

    count.times do

      thing_uuid = UUIDTools::UUID.random_create.to_s
      Thing.create :title =>"New Thing (#{thing_uuid})", :description =>
"Description for thing #{thing_uuid}"
    end

  end

end
```

This worker create `n` number of things, where `n` is the value passed to the worker.

Create a controller for "Things":

```
$ rails g controller Thing
```

```
class ThingController < ApplicationController

  def new
    ThingWorker.perform_async(2)
    redirect_to '/thing'
  end

  def index
    @things = Thing.all
  end

end
```

Add a view to inspect our collection of "Things":

```
$ mkdir app/views/things
$ touch app/views/things/index.html.erb
```

```
nil
```

Deploy the App

This app needs to be deployed twice for it to work, once as a Rails web app and once as a standalone Ruby app. One way to do this is to keep separate

Cloud Foundry manifests for each app type:

Web Manifest: Save this as `web-manifest.yml` :

```
---
applications:
- name: sidekiq
  memory: 256M
  instances: 1
  host: sidekiq
  domain: ${target-base}
  path: .
  services:
  - sidekiq-mysql:
  - sidekiq-redis:
```

Worker Manifest: Save this as `worker-manifest.yml` :

```
---
applications:
- name: sidekiq-worker
  memory: 256M
  instances: 1
  path: .
  command: bundle exec sidekiq
  no-route: true
  services:
  - sidekiq-redis:
  - sidekiq-mysql:
```

Since the url "sidekiq.cloudfoundry.com" is probably already taken, change it in `web-manifest.yml` first, then push the app with both manifest files:

```
$ cf push -f web-manifest.yml
$ cf push -f worker-manifest.yml
```

If the cf CLI asks for a URL for the worker app, select **none**.

Test the App

Test the app by visiting the new action on the "Thing" controller at the assigned url. In this example, the URL would be `http://sidekiq.cloudfoundry.com/thing/new` .

This creates a new Sidekiq job which is queued in Redis, then picked up by the worker app. The browser is then redirected to `/thing` which shows the collection of "Things".

Scale Workers

Use the `cf scale` command to change the number of Sidekiq workers.

Example:

```
$ cf scale sidekiq-worker -i 2
```

Use rails_serve_static_assets on Rails 4

By default Rails 4 returns a 404 if an asset is not handled via an external proxy such as Nginx. The `rails_serve_static_assets` gem enables your Rails server to deliver static assets directly, instead of returning a 404. You can use this capability to populate an edge cache CDN or serve files directly from your web app. The gem enables this behavior by setting the `config.serve_static_assets` option to `true` , so you do not need to configure it manually.

Add Custom Libraries

If your app requires external shared libraries that are not provided by the rootfs or the buildpack, you must place the libraries in an `ld_library_path` directory at the app root.

Note: You must keep these libraries up-to-date. They do not update automatically.

The Ruby buildpack automatically adds the directory `<app-root>/ld_library_path` to `LD_LIBRARY_PATH` so that your app can access these libraries at runtime.

Environment Variables

You can access environments variable programmatically. For example, you can obtain `VCAP_SERVICES` as follows:

```
ENV['VCAP_SERVICES']
```

Environment variables available to you include both those defined by the system and those defined by the Ruby buildpack, as described below. For more information about system environment variables, see the Application-Specific System Variables section of the *Cloud Foundry Environment Variables* topic.

BUNDLE_BIN_PATH

Location where Bundler installs binaries.

Example: `BUNDLE_BIN_PATH:/home/vcap/app/vendor/bundle/ruby/1.9.1/gems/bundler-1.3.2/bin/bundle`

BUNDLE_GEMFILE

Path to app Gemfile.

Example: `BUNDLE_GEMFILE:/home/vcap/app/Gemfile`

BUNDLE_WITHOUT

The `BUNDLE_WITHOUT` environment variable instructs Cloud Foundry to skip gem installation in excluded groups.

Use this with Rails applications, where “assets” and “development” gem groups typically contain gems that are not needed when the app runs in production.

Example: `BUNDLE_WITHOUT=assets`

DATABASE_URL

Cloud Foundry examines the `database_uri` for bound services to see if they match known database types. If known relational database services are bound to the app, the `DATABASE_URL` environment variable is set using the first match in the list.

If your app depends on `DATABASE_URL` to be set to the connection string for your service and Cloud Foundry does not set it, use the `cf set-env` command to can set this variable manually.

Example:

```
$ cf set-env my-app-name DATABASE_URL mysql://example-database-connection-string
```

GEM_HOME

Location where gems are installed.

Example: `GEM_HOME:/home/vcap/app/vendor/bundle/ruby/1.9.1`

GEM_PATH

Location where gems can be found.

Example: `GEM_PATH=/home/vcap/app/vendor/bundle/ruby/1.9.1:`

RACK_ENV

This variable specifies the Rack deployment environment. Valid value are `development`, `deployment`, and `none`. This governs which middleware is loaded to run the app.

Example: `RACK_ENV=development`

RAILS_ENV

This variable specifies the Rails deployment environment. Valid value are `development`, `test`, and `production`. This controls which of the environment-specific configuration files governs how the app is executed.

Example: `RAILS_ENV=production`

RUBYOPT

This Ruby environment variable defines command-line options passed to Ruby interpreter.

Example: `RUBYOPT: -I/home/vcap/app/vendor/bundle/ruby/1.9.1/gems/bundler-1.3.2/lib -rbundler/setup`

Getting Started Deploying Ruby Apps

Page last updated:

This topic provides links to additional information about getting started deploying apps using the Ruby buildpack. See the following topics for deployment guides specific to your app language and framework:

- Ruby
- Ruby on Rails

Getting Started Deploying Ruby Apps

Page last updated:

This guide is intended to walk you through deploying a Ruby app to Pivotal Application Service. If you experience a problem following the steps below, check the Known Issues topic, or refer to the Troubleshooting Application Deployment and Health topic.

Sample App Step
If you want to go through this tutorial using the sample app, run `git clone https://github.com/cloudfoundry-samples/pong_matcher_ruby.git` to clone the `pong_matcher_ruby` app from GitHub, and follow the instructions in the Sample App Step sections.

Note: Ensure that your Ruby app runs locally before continuing with this procedure.

Deploy a Ruby Application

This section describes how to deploy a Ruby application to PAS, and uses output from a sample app to show specific steps of the deployment process.

Prerequisites

- A Ruby 2.x application that runs locally on your workstation
- Bundler configured on your workstation
- Basic to intermediate Ruby knowledge
- The Cloud Foundry Command Line Interface (cf CLI) installed on your workstation

Step 1: Create and Bind a Service Instance for a Ruby Application

This section describes using the cf CLI to configure a Redis Cloud managed service instance for an app. You can use either the CLI or Apps Manager to perform this task.

PAS supports two types of service instances:

- Managed services integrate with PAS through service brokers that offer services and plans and manage the service calls between PAS and a service provider.
- User-provided service instances enable you to connect your application to pre-provisioned external service instances.

For more information about creating and using service instances, refer to the Services Overview topic.

Create a Service Instance

Run `cf marketplace` to view managed and user-provided services and plans that are available to you.

The example shows three of the available managed database-as-a-service providers and the plans that they offer: `cleardb` MySQL and `elephantsql` PostgreSQL as a Service.

```
$ cf marketplace
Getting services from marketplace in org Cloud-Apps / space development as clouduser@example.com...
OK

service        plans      description
...
cleardb        spark, boost, amp, shock      Highly available MySQL for your Apps
...
elephantsql     turtle, panda, hippo, elephant   PostgreSQL as a Service
...
```

Run `cf create-service SERVICE PLAN SERVICE_INSTANCE` to create a service instance for your app. Choose a SERVICE and PLAN from the list, and provide a unique name for the SERVICE_INSTANCE.

Sample App Step
Run `cf create-service rediscloud 30mb redis`. This creates a service instance named `redis` that uses the `rediscloud` service and the `30mb` plan, as the example below shows.

```
$ cf create-service rediscloud 30mb redis
Creating service redis in org Cloud-Apps / space development as clouduser@example.com....
OK
```

Bind a Service Instance

When you bind an app to a service instance, PAS writes information about the service instance to the VCAP_SERVICES app environment variable. The app can use this information to integrate with the service instance.

Most services support bindable service instances. Refer to your service provider's documentation to confirm if they support this functionality.

You can bind a service to an application with the command `cf bind-service APPLICATION SERVICE_INSTANCE`.

Alternately, you can configure the deployment manifest file by adding a `services` block to the `applications` block and specifying the service instance. For more information and an example on service binding using a manifest, see the Sample App Step.

You can also bind a service using the Apps Manager.

Sample App Step
You can skip this step. The manifest for the sample app contains a `services` sub-block in the `applications` block, as the example below shows. This binds the `redis` service instance that you created in the previous step.

```
services:
  - redis
```

Step 2: Configure Deployment Options

Configure the Deployment Manifest

You can specify app deployment options in a manifest that the `cf push` command uses. For more information about application manifests and supported attributes, refer to the Deploying with Application Manifests topic.

Configure a Production Server

PAS uses the default standard Ruby web server library, WEBrick, for Ruby and RoR apps. However, PAS can support a more robust production web server, such as Phusion Passenger, Puma, Thin, or Unicorn. If your app requires a more robust web server, refer to the Configuring a Production Server topic for help configuring a server other than WEBrick.

Sample App Step
You can skip this step. The `manifest.yml` file for `pong_matcher_ruby` does not require any additional configuration to deploy the app.

Step 3: Log in and Target the API Endpoint

Run `cf login -a API_ENDPOINT`, enter your login credentials, and select a space and org. The API endpoint is the URL of the Cloud Controller in your PAS instance.

Sample App Step
You must do this step to run the sample app.

Step 4: Deploy an App

Note: You must use the cf CLI to deploy apps.

From the root directory of your application, run `cf push APP_NAME` to deploy your application.

`cf push APP_NAME` creates a URL route to your application in the form HOST.DOMAIN, where HOST is your APP_NAME and DOMAIN is specified by your administrator. Your DOMAIN is `shared-domain.example.com`. For example: `cf push my-app` creates the URL `my-app.shared-domain.example.com`.

The URL for your app must be unique from other apps that PAS hosts or the push will fail. Use the following options to help create a unique URL:

- `-n` to assign a different HOST name for the app.
- `--random-route` to create a URL that includes the app name and random words.
- `cf help push` to view other options for this command.

If you want to view log activity while the app deploys, launch a new terminal window and run `cf logs APP_NAME`.

Once your app deploys, browse to your app URL. Search for the `urls` field in the `App started` block in the output of the `cf push` command. Use the URL to access your app online.

Sample App Step
Run `cf push pong_matcher_ruby -n HOST_NAME`.

Example: `cf push pong_matcher_ruby -n pongmatch-ex12`

The example below shows the terminal output of deploying the `pong_matcher_ruby` app. `cf push` uses the instructions in the manifest file to create the app, create and bind the route, and upload the app. It then binds the app to the `redis` service and follows the instructions in the manifest to start one instance of the app with 256M. After the app starts, the output displays the health and status of the app.

Note: The `pong_matcher_ruby` app does not include a web interface. To interact with the `pong_matcher_ruby` app, see the interaction instructions on GitHub: https://github.com/cloudfoundry-samples/pong_matcher_ruby.

```
$ cf push pong_matcher_ruby -n pongmatch-ex12
Using manifest file /Users/clouduser/workspace/pong_matcher_ruby/manifest.yml

Creating app pong_matcher_ruby in org Cloud-Apps / space development as clouduser@example.com...
OK

Creating route pongmatch-ex12.shared-domain.example.com
    Binding pongmatch-ex12.shared-domain.example.com to pong_matcher_ruby...
OK

Uploading pong_matcher_ruby...
Uploading app files from: /Users/clouduser/workspace/pong_matcher_ruby
Uploading 8.8K, 12 files
OK
Binding service redis to app pong_matcher_ruby in org Cloud-Apps / space development as clouduser@example.com...
OK

Starting app pong_matcher_ruby in org Cloud-Apps / space development as clouduser@example.com...
OK
...

0 of 1 instances running, 1 starting
1 of 1 instances running

App started

Showing health and status for app pong_matcher_ruby in org Cloud-Apps / space development as clouduser@example.com...
OK

requested state: started
instances: 1/1
usage: 256M x 1 instances
urls: pongmatch-ex12.cfapps.io

     state     since                    cpu    memory         disk
#0   running   2014-12-09 10:04:40 AM   0.0%   35.2M of 256M   45.8M of 1G
```

Step 5: Test a Deployed App

You've deployed an app to PAS!

Use the cf CLI or Apps Manager to review information and administer your app and your PAS account. For example, you could edit the `manifest.yml` to increase the number of app instances from 1 to 3, and redeploy the app with a new app name and host name.

See the Manage Your Application with the cf CLI section for more information. See also Using the Apps Manager.

Manage Your Application with the cf CLI

Run `cf help` to view a complete list of commands, grouped by task categories, and run `cf help COMMAND` for detailed information about a specific command. For more information about using the cf CLI, refer to the Cloud Foundry Command Line Interface (cf CLI) topics, especially the Getting Started with cf CLI topic.

> **Note**: You cannot perform certain tasks in the CLI or Apps Manager because these are commands that only a PAS administrator can run. If you are not a PAS administrator, the following message displays for these types of commands: `error code: 10003, message: You are not authorized to perform the requested action` For more information about specific Admin commands you can perform with the Apps Manager, depending on your user role, refer to the Getting Started with the Apps Manager topic.

Troubleshooting

If your application fails to start, verify that the application starts in your local environment. Refer to the Troubleshooting Application Deployment and Health topic to learn more about troubleshooting.

App Deploy Fails

Even when deploying an app fails, the app might exist on PAS. Run `cf apps` to review the apps in the currently targeted org and space. You might be able to correct the issue using the CLI or Apps Manager, or you might have to delete the app and redeploy.

Common reasons deploying an app fails include:

- You did not successfully create and bind a needed service instance to the app, such as a PostgreSQL service instance. Refer to Step 2: Create and Bind a Service Instance for a Ruby Application.
- You did not successfully create a unique URL for the app. Refer to the troubleshooting tip **App Requires Unique URL**.

App Requires Unique URL

PAS requires that each app that you deploy has a unique URL. Otherwise, the new app URL collides with an existing app URL and PAS cannot successfully deploy the app. You can resolve this issue by running `cf push` with either of the following flags to create a unique URL:

- `-n` to assign a different HOST name for the app.
- `--random-route` to create a URL that includes the app name and random words. Using this option might create a long URL, depending on the number of words that the app name includes.

Getting Started Deploying Ruby on Rails Apps

Page last updated:

This guide walks you through deploying a Ruby on Rails app to Pivotal Cloud Foundry (PCF).

Prerequisites

Before you deploy your app to PCF, ensure you have the following:

- A Ruby on Rails app that runs locally
- Bundler installed on your machine
- Your username and password for the targeted Cloud Foundry instance that have Space Developer permissions
- The Cloud Foundry Command Line Interface (cf CLI)

If you want to follow this tutorial using a sample app and a managed PostgreSQL service, clone the `rails-sample` app from GitHub by running the following command:

```
$ git clone https://github.com/cloudfoundry-samples/rails_sample_app
```

Step 1: Log in to Cloud Foundry

To log in to your Cloud Foundry instance and target its API endpoint, do the following:

1. Run the `cf login -a API-URL` command, where `API-URL` is the API endpoint for the instance. For more information, see the Identifying the API Endpoint for your PAS Instance topic.
2. Enter your credentials to log in and to select a Space and Org.

Step 2: (Optional) Create Service Instances

Note: If you use the sample Ruby on Rails app to go through this tutorial, Step 2 is not optional.

An app that uses services, such as a database, messaging, or email server, is not fully functional until you provision the services and, if required, bind the services to the app. You can use either the CLI or Apps Manager to perform this task.

You can provision service instances for an app that has already been pushed to Cloud Foundry. For more information about creating and using service instances, refer to the Services Overview topic.

Create a Service Instance

To create a service instance:

1. Confirm service availability in the Marketplace:

   ```
   $ cf marketplace
   ```

2. Run the `cf create-service SERVICE-NAME PLAN-NAME SERVICE-INSTANCE-NAME` command. See the following example for the sample Ruby on Rails app:

   ```
   $ cf create-service elephantsql turtle rails-postgres
   Creating service rails-postgres in org my-org / space development as user@example.com....
   OK
   ```

   In the command above, `elephantsql` is the name of the service, `turtle` is the name of the service plan, and `rails-postgres` is the unique name you gave your service instance.

Pivotal

Bind a Service Instance

When you bind a service instance to an app, Cloud Foundry writes information about the service instance to the VCAP_SERVICES environment variable. The app can use this information to integrate with the service instance.

Most services support bindable service instances. Refer to your service provider's documentation to confirm whether they support this functionality. To bind a service instance to your app, you can do one of the following:

- Specify your service instance in the `services` sub-block of the app manifest file, which enables Cloud Foundry to bind the service instance to the app during deployment. See the following excerpt from the manifest file of the sample Ruby on Rails app:

```
services:
  - rails-postgres
```

- Bind a service instance after pushing your app to Cloud Foundry:

```
$ cf bind-service APP-NAME SERVICE-INSTANCE-NAME
```

Note: You must restart or in some cases re-push your app for changes to be applied to the `VCAP_SERVICES` environment variable and for the app to recognize these changes.

Step 3: Define Deployment Options

You can define deployment options on the command line, in a manifest file, or both together.

(Optional) Configure the Deployment Manifest

You can specify app deployment options in a manifest that the `cf push` command uses. See the manifest file for the sample Ruby on Rails app:

```
---
applications:
- name: rails-sample
  memory: 256M
  instances: 1
  path: .
  command: bundle exec rake db:migrate && bundle exec rails s -p $PORT
  services:
    - rails-postgres
```

For more information about app manifests and supported attributes, refer to the Deploying with Application Manifests topic.

(Optional) Configure a Production Server

For Ruby and Ruby on Rails apps, PCF uses WEBrick, the default standard Ruby web server library. However, PCF can support a more robust production web server, such as Phusion Passenger, Puma, Thin, or Unicorn. If your app requires a more robust web server, refer to the Configuring a Production Server topic for help with configuring a server other than WEBrick.

Step 4: Deploy a Ruby on Rails App

To deploy your app, run `cf push APP-NAME` from the root directory of the app.

The `cf push APP-NAME` command creates a URL route to your app in the `HOST.DOMAIN` format, where `HOST` is your `APP-NAME` and `DOMAIN` is specified by your administrator. For example, for the `shared-domain.example.com` domain, running `cf push APP-NAME` creates the `APP-NAME.shared-domain.example.com` URL.

If the URL for your app is not unique, the deployment fails. For more information about creating unique URLs, see the Configure Domains section of the *Deploy an Application* topic.

See the following example for a sample Ruby on Rails app:

```
$ cf push rails-sample
Using manifest file ~/workspace/rails_sample_app/manifest.yml

Updating app rails-sample in org Cloud-Apps / space development as clouduser@example.com...
OK

Using route rails-sample.shared-domain.example.com
Uploading rails-sample...
Uploading app files from: ~/workspace/rails_sample_app
Uploading 445.7K, 217 files
OK
Binding service rails-postgres to app rails-sample in org Cloud-Apps / space development as clouduser@example.com...
OK

Starting app rails-sample in org Cloud-Apps / space development as clouduser@example.com...
OK

...

0 of 1 instances running, 1 starting
1 of 1 instances running

App started

Showing health and status for app rails-sample in org Cloud-Apps / space development as clouduser@example.com...
OK

requested state: started
instances: 1/1
usage: 256M x 1 instances
urls: rails-sample.shared-domain.example.com

     state     since                    cpu    memory         disk
#0   running   2014-08-25 03:32:10 PM   0.0%   68.4M of 256M   73.4M of 1G
```

The example above shows that the `cf push` command uses the instructions in the manifest file to create the app, create and bind the route, and upload the app. It then binds the app to the `rails-postgres` service and starts one instance of the app with 256 MB of RAM. After the app starts, the output displays the health and status of the app.

> **Note**: If you want to view log activity while the app is deployed, launch a new terminal window and run `cf logs APP-NAME`.

You can verify that your app is running by browsing to the URL generated in the output of the `cf push` command.

Step 5: (Optional) Configure Service Connections

For Ruby on Rails versions 4.0.0 and earlier, the Ruby buildpack overwrites the `config/database.yml` file using the contents of the `DATABASE_URL` environment variable. This variable can be set by the user, or it will be automatically set based on bound service instances.

For Ruby on Rails versions 4.1.0 and later, the Ruby buildpack does not modify the `config/database.yml` file.

When the app starts, the standard Ruby on Rails behavior for `DATABASE_URL` and `config/database.yml` applies.

What to Do Next

You have deployed an app to PCF. Consult the sections below for information about what to do next.

Test a Deployed App

Use the cf CLI or Apps Manager to review information and administer your app and your PCF account. For example, you could edit the `manifest.yml` file to increase the number of app instances from 1 to 3 or redeploy the app with a new app name.

Manage Your App with the cf CLI

Run `cf help` to view a complete list of commands and run `cf help COMMAND` for detailed information about a specific command. For more information about using the cf CLI, refer to the cf CLI topics, especially the Getting Started with cf CLI topic.

Troubleshooting

If your app fails to start, verify that the app starts in your local environment. Refer to the Troubleshooting Application Deployment and Health topic to learn more about troubleshooting.

Configure Rake Tasks for Deployed Apps

Page last updated:

For PAS to automatically invoke a Rake task while a Ruby or Ruby on Rails app is deployed, you must do the following:

- Include the Rake task in your app
- Configure the application start command using the `command` attribute in the application manifest

The following is an example that shows how to invoke a Rake database migration task at application startup.

1. Create a file with the Rake task name and the extension `.rake`, and store it in the `lib/tasks` directory of your application.

2. Add the following code to your rake file:

```
namespace :cf do
    desc "Only run on the first application instance"
    task :on_first_instance do
        instance_index = JSON.parse(ENV["VCAP_APPLICATION"])["instance_index"] rescue nil
    exit(0) unless instance_index == 0
    end
end
```

   This Rake task limits an idempotent command to the first instance of a deployed application.

3. Add the task to the `manifest.yml` file with the `command` attribute, referencing the idempotent command `rake db:migrate` chained with a start command.

```
applications:
- name: my-rails-app
  command: bundle exec rake cf:on_first_instance db:migrate && rails s -p $PORT
```

Environment Variables Defined by the Ruby Buildpack

Pivotal Application Service provides configuration information to apps through environment variables. This topic describes the additional environment variables provided by the Ruby buildpack.

For more information about the standard environment variables provided by Pivotal Application Service, see the Cloud Foundry Environment Variables topic.

Ruby Buildpack Environment Variables

The following table describes the environment variables provided by the Ruby buildpack.

Environment Variable	Description
`BUNDLE_BIN_PATH`	The directory where Bundler installs binaries. Example: `BUNDLE_BIN_PATH:/home/vcap/app/vendor/bundle/ruby/1.9.1/gems/bundler-1.3.2/bin/bundle`
`BUNDLE_GEMFILE`	The path to the Gemfile for the app. Example: `BUNDLE_GEMFILE:/home/vcap/app/Gemfile`
`BUNDLE_WITHOUT`	Instructs Cloud Foundry to skip gem installation in excluded groups. Use this with Rails applications, where "assets" and "development" gem groups typically contain gems that are not needed when the app runs in production. Example: `BUNDLE_WITHOUT=assets`
`DATABASE_URL`	Cloud Foundry examines the `database_uri` for bound services to see if they match known database types. If known relational database services are bound to the app, then the `DATABASE_URL` environment variable is set to the first services in the list. If your application requires that `DATABASE_URL` is set to the connection string for your service, and Cloud Foundry does not set it, use the Cloud Foundry Command Line Interface (cf CLI) `cf set-env` command to set this variable manually. Example: `$ cf set-env my-app DATABASE_URL mysql://example-database-connection-string`
`GEM_HOME`	The directory where gems are installed. Example: `GEM_HOME:/home/vcap/app/vendor/bundle/ruby/1.9.1`
`GEM_PATH`	The directory where gems can be found. Example: `GEM_PATH=/home/vcap/app/vendor/bundle/ruby/1.9.1:`
`RACK_ENV`	The Rack deployment environment, which governs the middleware loaded to run the app. Valid value are `development`, `deployment`, and `none`. Example: `RACK_ENV=none`
`RAILS_ENV`	The Rails deployment environment, which controls which environment-specific configuration file governs how the app is executed. Valid value are `development`, `test`, and `production`. Example: `RAILS_ENV=production`
`RUBYOPT`	Defines command-line options passed to Ruby interpreter. Example: `RUBYOPT: -I/home/vcap/app/vendor/bundle/ruby/1.9.1/gems/bundler-1.3.2/lib -rbundler/setup`

Configure Service Connections for Ruby

Page last updated:

After you create a service instance and bind it to an application, you must configure the application to connect to the service.

Query VCAP_SERVICES with cf-app-utils

The `cf-apps-utils` gem allows your application to search for credentials from the `VCAP_SERVICES` environment variable by name, tag, or label.

- cf-app-utils-ruby

VCAP_SERVICES defines DATABASE_URL

At runtime, Cloud Foundry creates a `DATABASE_URL` environment variable for every application based on the VCAP_SERVICES environment variable.

Example VCAP_SERVICES:

```
VCAP_SERVICES =
{
  "elephantsql": [
    {
      "name": "elephantsql-c6c60",
      "label": "elephantsql",
      "credentials": {
        "uri": "postgres://exampleuser:examplepass@babar.elephantsql.com:5432/exampledb"
      }
    }
  ]
}
```

Based on this `VCAP_SERVICES`, Cloud Foundry creates the following `DATABASE_URL` environment variable:

```
DATABASE_URL = postgres://exampleuser:examplepass@babar.elephantsql.com:5432/exampledb
```

Cloud Foundry uses the structure of the `VCAP_SERVICES` environment variable to populate `DATABASE_URL`. Any service containing a JSON object with the following form will be recognized by Cloud Foundry as a candidate for `DATABASE_URL`:

```
{
  "some-service": [
    {
      "credentials": {
        "uri": "<some database URL>"
      }
    }
  ]
}
```

Cloud Foundry uses the first candidate found to populate `DATABASE_URL`.

Configure Non-Rails Applications

Non-Rails applications can also access the `DATABASE_URL` variable.

If you have more than one service with credentials, only the first will be populated into `DATABASE_URL`. To access other credentials, you can inspect the `VCAP_SERVICES` environment variable.

```
vcap_services = JSON.parse(ENV['VCAP_SERVICES'])
```

Use the hash key for the service to obtain the connection credentials from `VCAP_SERVICES`.

- For services that use the v2 format, the hash key is the name of the service.
- For services that use the v1 format, the hash key is formed by combining the service provider and version, in the format PROVIDER-VERSION.

For example, the service provider “p-mysql” with version “n/a” forms the hash key `p-mysql-n/a`.

Seed or Migrate Database

Before you can use your database the first time, you must create and populate or migrate it. For more information, see Migrating a Database in Cloud Foundry.

Troubleshooting

To aid in troubleshooting issues connecting to your service, you can examine the environment variables and log messages Cloud Foundry records for your application.

View Environment Variables

Use the `cf env` command to view the Cloud Foundry environment variables for your application. `cf env` displays the following environment variables:

- The `VCAP_SERVICES` variables existing in the container environment
- The user-provided variables set using the `cf set-env` command

```
$ cf env my-app
Getting env variables for app my-app in org My-Org / space development as admin...
OK

System-Provided:
{
 "VCAP_SERVICES": {
  "p-mysql-n/a": [
   {
    "credentials": {
      "uri":"postgres://lrra:e6B-X@p-mysqlprovider.example.com:5432/lraa
    },
    "label": "p-mysql-n/a",
    "name": "p-mysql",
    "syslog_drain_url": "",
    "tags": ["postgres","postgresql","relational"]
   }
  ]
 }
}

User-Provided:
my-env-var: 100
my-drain: http://drain.example.com
```

View Logs

Use the `cf logs` command to view the Cloud Foundry log messages for your application. You can direct current logging to standard output, or you can dump the most recent logs to standard output.

Run `cf logs APPNAME` to direct current logging to standard output:

```
$ cf logs my-app
Connected, tailing logs for app my-app in org My-Org / space development as admin...
1:27:19.72 [App/0]  OUT [CONTAINER]  org.apache.coyote.http11.Http11Protocol  INFO  Starting ProtocolHandler ["http-bio-61013"]
1:27:19.77 [App/0]  OUT [CONTAINER]   org.apache.catalina.startup.Catalina     INFO  Server startup in 10427 ms
```

Run `cf logs APPNAME --recent` to dump the most recent logs to standard output:

```
$ cf logs my-app --recent
Connected, dumping recent logs for app my-app in org My-Org / space development as admin...
1:27:15.93 [App/0]  OUT 15,935  INFO EmbeddedDatabaseFactory:124 - Creating embedded database 'SkyNet'
1:27:16.31 [App/0]  OUT 16,318  INFO LocalEntityManagerFactory:287 - Building TM container EntityManagerFactory for unit 'default'
1:27:16.50 [App/0]  OUT 16,505  INFO Version:37 - HCANN001: Hibernate Commons Annotations {4.0.1.Final}
1:27:16.51 [App/0]  OUT 16,517  INFO Version:41 - HHH412: Hibernate Core {4.1.9.Final}
1:27:16.95 [App/0]  OUT 16,957  INFO SkyNet-Online:73 - HHH268: Transaction strategy: org.hibernate.internal.TransactionFactory
1:27:16.96 [App/0]  OUT 16,963  INFO InintiateTerminatorT1000Deployment:48 - HHH000397: Using TranslatorFactory
1:27:17.02 [App/0]  OUT 17,026  INFO Version:27 - HV001: Hibernate Validator 4.3.0.Final
```

If you encounter the error, “A fatal error has occurred. Please see the Bundler troubleshooting documentation,” update your version of bundler and run `bundle install`.

```
$ gem update bundler
$ gem update --system
$ bundle install
```

Support for Windows Gemfiles

Page last updated:

This topic describes how the Ruby buildpack handles dependencies on Windows machines.

Windows Gemfiles

When a `Gemfile.lock` is generated on a Windows machine, it often contains gems with Windows-specific versions. This results in versions of gems such as `mysql2`, `thin`, and `pg` containing “-x86-mingw32.” For example, the `Gemfile` may contain the following:

```
gem 'sinatra'
gem 'mysql2'
gem 'json'
```

When you run `bundle install` with the above `Gemfile` on a Windows machine, it results in the following `Gemfile.lock`:

```
GEM remote: http://rubygems.org/
specs:
  json (1.7.3)
  mysql2 (0.3.11-x86-mingw32)
  rack (1.4.1)
  rack-protection (1.2.0)
  rack sinatra (1.3.2)
  rack (~> 1.3, >= 1.3.6)
  rack-protection (~> 1.2)
  tilt (~> 1.3, >= 1.3.3)
  tilt (1.3.3)
PLATFORMS x86-mingw32
DEPENDENCIES
  json
  mysql2
  sinatra
```

Notice the “-x86-mingw32” in the version number of `mysql2`. Since Cloud Foundry runs on Linux machines, this would fail to install. To mitigate this, the Ruby Buildpack removes the `Gemfile.lock` and uses Bundler to resolve dependencies from the `Gemfile`.

Note: Removing the `Gemfile.lock` will cause dependency versions to be resolved from the `Gemfile`. This could result in different versions being installed than those listed in the `Gemfile.lock`.

Staticfile Buildpack

Page last updated:

Overview

This topic describes how to configure the Staticfile buildpack and use it to push static content to the web. It also shows you how to serve a simple "Hello World" page using the Staticfile buildpack.

> **Note**: BOSH configured custom trusted certificates are not supported by the Staticfile buildpack.

Definitions

Staticfile app: An app or content that requires no backend code other than the NGINX webserver, which the buildpack provides. Examples of staticfile apps are front-end JavaScript apps, static HTML content, and HTML/JavaScript forms.

Staticfile buildpack: The buildpack that provides runtime support for staticfile apps and apps with backends hosted elsewhere. To find which version of NGINX the current Staticfile buildpack uses, see the Staticfile buildpack release notes.

Staticfile Requirement

Pivotal Application Service requires a file named `Staticfile` in the root directory of the app to use the Staticfile buildpack with the app.

Memory Usage

NGINX requires 20 MB of RAM to serve static assets. When using the Staticfile buildpack, we recommend pushing apps with the `-m 64M` option to reduce RAM allocation from the default 1 GB allocated to containers by default.

"Hello World" Tutorial

Follow the procedure below to create and push a single page app using the Staticfile buildpack.

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1</td><td>Create and move into a root directory for the sample app in your workspace:<pre>$ mkdir sample
$ cd sample</pre></td></tr>
<tr><td>2</td><td>Create an <code>index.html</code> file that contains some text:<pre>$ echo 'Hello World' > index.html</pre></td></tr>
<tr><td>3</td><td>Create an empty file named <code>Staticfile</code>:<pre>$ touch Staticfile</pre></td></tr>
<tr><td>4</td><td>Use the <code>cf login</code> command to log in to PAS.
For more information, see the Login section of the Getting Started with the cf CLI documentation.<pre>$ cf login</pre></td></tr>
<tr><td>5</td><td>Push the sample app:<pre>$ cf push hello -m 64M</pre></td></tr>
</table>

6	Find the URL of the app in the output.
	A fragment of output is shown below:
	`Creating app hello in org sample-org / space sample-space as username@example.com...`
	`OK`
	`…`
	`requested state: started`
	`instances: 1/1`
	`usage: 64M x 1 instances`
	`urls: hello.example.com`

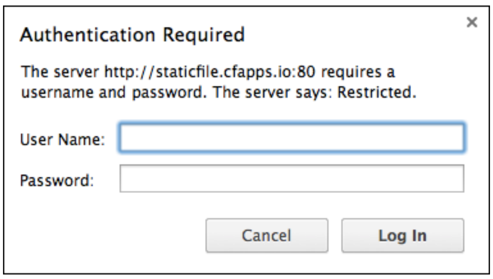
7	Navigate to the URL to see the sample app running.

Configuring the Buildpack and Pushing the App

This section describes ways you can configure the Staticfile buildpack options and how to push your Staticfile app.

Configuration Options

This table describes aspects of the Staticfile buildpack that you can configure using the Configure Your App details in the next table.

Options	Description
Alternative root	Allows you to specify a root directory other than the default, which is `index.html`. For example, you can specify that the buildpack serves index.html and all other assets from an alternate folder where your HTML/CSS/JavaScript files exist, such as `dist/` or `public/`.
Directory list	An HTML page that displays a directory index for your site. A sample of a directory list is shown below.  If your site is missing an index.html, your app displays a directory list instead of the standard 404 error page.
SSI	Server Side Includes (SSI) allows you to show the contents of files on a web page on the web server. For general information about SSI, see the Server Side Includes entry on Wikipedia.
Pushstate routing	Keeps browser-visible URLs clean for client-side JavaScript apps that serve multiple routes. For example, pushstate routing allows a single JavaScript file route to multiple anchor-tagged URLs that look like `/some/path1` instead of `/some#path1`.
GZip file serving and compressing	The gzip_static and gunzip modules are enabled by default. This allows NGINX to serve files stored in compressed GZ format, and to uncompresses them for clients that do not support compressed content or responses. You may want to disable compression in particular circumstances, for example if serving to very old browser clients.
Basic authentication	Allows you to place simple access controls on your app. 

Proxy support	Allows you to use a proxy to download dependencies during staging.
Force HTTPS	A way to enforce that all requests are sent through HTTPS. This redirects non-HTTPS requests as HTTPS requests. For an example of when to avoid forcing HTTPS, see About FORCE_HTTPS with Reverse Proxies.
Dot Files	By default, hidden files (those starting with a `.`) are not served by this buildpack.
HTTP Strict Transport Security	Causes NGINX to respond to all requests with the header `Strict-Transport-Security: max-age=31536000`. This forces browsers to make all subsequent requests over HTTPS. Defaults to a max-age of one year.
HTTP Strict Transport Security Include SubDomains	Causes NGINX to respond to all requests with the header `Strict-Transport-Security: max-age=31536000; includeSubDomains`. This forces browsers to make all subsequent requests over HTTPS including subdomains. Defaults to a max-age of one year.
HTTP Strict Transport Security Preload	Causes NGINX to respond to all requests with the header `Strict-Transport-Security: max-age=31536000; includeSubDomains; preload`. This forces browsers to make all subsequent requests over HTTPS including subdomains and requests inclusion in browser-managed HSTS preload lists (see https://hstspreload.org). Defaults to a max-age of one year.

Custom Location configuration	To customize the `location` block of the NGINX configuration file, follow these steps:
	1. Set a `root` directory (`location_include` only works in conjunction with `root`)
	2. Create a file with location scoped NGINX directives. See the following example, which causes visitors of your site to receive the `X-MySiteName` HTTP header:
	File: `nginx/conf/includes/custom_header.conf`
	Content: `add_header X-MySiteName BestSiteEver;`
	3. Set the `location_include` variable in your **Staticfile** to the path of the file from the previous step:
	`...`
	`root: public`
	`location_include: includes/*.conf`
	`...`
	For information about NGINX directives, see NGINX documentation.

Custom NGINX configuration	Additional NGINX configuration can be applied, such as mapping file extensions to mime types. See the NGINX documentation for examples.

Configure Your App

1. In the root directory of your app, create an empty file named `Staticfile`:

```
$ touch Staticfile
```

2. Configure the Staticfile buildpack for the needs of your app.

Note: This setting will default to http_strict_transport_security_include_subdomains being true.

If you want to...	Then...	More information
serve `index.html` and other assets from a location other than the root directory	add this line to the `Staticfile`: `root: YOUR-DIRECTORY-NAME` For example, `root: public`	Alternative root
display a directory list instead of the standard 404 error	add this line to the `Staticfile`: `directory: visible`	Directory list
enable SSI	add this line to the `Staticfile`: `ssi: enabled`	SSI
enable pushstate routing	add this line to the `Staticfile`:	Pushstate

	`pushstate: enabled`	routing
disable gzip_static and gunzip modules	add this line to the `Staticfile`: `gzip: off`	GZip

enable basic authentication for your app or website	1. Create a hashed username and password pair for each user, using a site like Htpasswd Generator
	2. Create a file named `Staticfile.auth` in the root directory or alternative root directory, and add one or more user/password lines to it. For example,
	`bob:$apr1$DuUQEQp8$ZccZCHQElNSjrgerwSFC0`
	`alice:$apr1$4IRQGcD/$UMFLnIHSD9ZHJ86TR4zx`
		Basic authentication

use a proxy for downloading dependencies during staging	set the `http_proxy` and `https_proxy` environment variables.	Proxy support
force HTTPS	set the `FORCE_HTTPS` environment variable to `true`. **Note**: Do not enable `FORCE_HTTPS` if you have a proxy server or load balancer that terminates SSL/TLS. Doing so can cause infinite redirect loops, for example, if you use Flexible SSL with CloudFlare.	Force HTTPS
host dot files	To enable serving hidden (dot files), use `host_dot_files: true` in the `Staticfile`.	Dot Files
force the receiving browser to make subsequent requests over HTTPS	add this line to the `Staticfile`: `http_strict_transport_security: true` **Note**: Because this setting persists in browsers for a long time, only enable this setting after ensuring you have completed your configuration.	HTTP Strict Transport Security
force the receiving browser to make subsequent requests over HTTPS for all sub-domains	add this line to the `Staticfile`: `http_strict_transport_security_include_subdomains: true` **Note**: This setting will default to http_strict_transport_security being true.	HTTP Strict Transport Security
force the receiving browser to make subsequent requests over HTTPS and add it to the Chrome's HSTS Preload list	add this line to the `Staticfile`: `http_strict_transport_security_preload: true` **Note**: This setting will default to http_strict_transport_security being true.	HTTP Strict Transport Security
make additional configuration changes to NGINX	add `nginx.conf` and `mime.types` files to your root folder, or to the alternate root folder if you specified one.	NGINX documentation

Push Your App

Follow the steps below to push your application.

Step	Action

1.	Use the `cf push APP_NAME -m 64M` command to push your app. Replace `APP_NAME` with the name you want to give your application. For example:
	`$ cf push my-app -m 64M`
	`Creating app my-app in org sample-org / space sample-space as username@example.com...`
	`OK`
	`...`
	`requested state: started`
	`instances: 1/1`
	`usage: 64M x 1 instances`
	`urls: my-app.example.com`
	Or, if you do not have the buildpack, or the installed version is out-of-date, use the `-b` option to specify the buildpack as follows:
	`cf push APP_NAME -b https://github.com/cloudfoundry/staticfile-buildpack.git`

2.	Find the URL of your app in the output from the push command and navigate to it to see your static app running.

Help and Support

A number of channels exist where you can get more help when using the Staticfile buildpack, or with developing your own Staticfile buildpack.

- **Staticfile Buildpack Repository in Github**: Find more information about using and extending the Staticfile buildpack in GitHub repository.
- **Release Notes**: Find current information about this buildpack on the Staticfile buildpack release page in GitHub.
- **Slack**: Join the #buildpacks channel in our Slack community.

Customizing and Developing Buildpacks

Page last updated:

Buildpacks enable you to packaging frameworks and/or runtime support for your application. Cloud Foundry provides with system buildpacks out-of-the-box and provides an interface for customizing existing buildpacks and developing new ones.

Customizing and Creating Buildpacks

If your application uses a language or framework that the Cloud Foundry system buildpacks do not support, do one of the following:

- Use a Cloud Foundry Community Buildpack.
- Use a Heroku Third-Party Buildpack.
- Customize an existing buildpack or create your own custom buildpack. A common development practice for custom buildpacks is to fork existing buildpacks and sync subsequent patches from upstream. For information about customizing an existing buildpack or creating your own, see the following:
  - Creating Custom Buildpacks
  - Packaging Dependencies for Offline Buildpacks

Maintaining Buildpacks

After you have modified an existing buildpack or created your own, it is necessary to maintain it. Refer to the following when maintaining your own buildpacks:

- Merging from Upstream Buildpacks
- Upgrading Dependency Versions

Note: To configure a production server for your web app, see the Configuring a Production Server topic.

Using CI for Buildpacks

For information about updating and releasing a new version of a Cloud Foundry buildpack through the Cloud Foundry Buildpacks Team Concourse pipeline, see Using CI for Buildpacks. You can use this as a model when working with Concourse to build and release new versions of your own buildpacks.

Pivotal

Creating Custom Buildpacks

Page last updated:

This topic describes how to create custom buildpacks for Pivotal Application Service (PAS).

For more information about how buildpacks work, see the Understanding Buildpacks topic.

Package Custom Buildpacks

PAS buildpacks can work with limited or no Internet connectivity. The Buildpack Packager RubyGem gives the same flexibility to custom buildpacks, enabling them to work in partially or completely disconnected environments.

Use the Buildpack Packager

1. Ensure that you have installed the Buildpack Packager RubyGem.

2. Create a manifest.yml in your buildpack.

3. Run the packager in cached mode:

```
$ buildpack-packager --cached
```

The packager will add (almost) everything in your buildpack directory into a zip file. It will exclude anything marked for exclusion in your manifest.

In cached mode, the packager downloads and adds dependencies as described in the manifest.

The packager has the following option flags:

- `--force-download`: By default, the packager stores the dependencies that it downloads while building a cached buildpack in a local cache at `~/.buildpack-packager`. Storing dependencies enables the packager to avoid re-downloading them when repackaging similar buildpacks. Running `buildpack-packager --cached` with the `--force-download` option forces the packager to download dependencies from the S3 host and ignore the local cache. When packaging an uncached buildpack, `--force-download` does nothing.
- `--use-custom-manifest`: To include a different manifest file in your packaged buildpack, call the packager with the `--use-custom-manifest PATH/TO/MANIFEST.YML` option. The packager generates a buildpack with the specified manifest. If you are building a cached buildpack, the packager vendors dependencies from the specified manifest as well.

For more information, see the documentation at the Buildpack Packager Github repository.

Use and Share the Packaged Buildpack

After you have packaged your buildpack using `buildpack-packager` you can use the resulting `.zip` file locally, or share it with others by uploading it to any network location that is accessible to the CLI. Users can then specify the buildpack with the `-b` option when they push apps. See Deploying Apps with a Custom Buildpack for details.

Note: Offline buildpack packages may contain proprietary dependencies that require distribution licensing or export control measures. For more information about offline buildpacks, refer to Packaging Dependencies for Offline Buildpacks.

You can also use the `cf create-buildpack` command to upload the buildpack into your Cloud Foundry deployment, making it accessible without the `-b` flag:

```
$ cf create-buildpack BUILDPACK PATH POSITION [--enable|--disable]
```

You can find more documentation in the Managing Custom Buildpacks topic.

Specify a Default Version

As of `buildpack-packager` version 2.3.0, you can specify the default version for a dependency by adding a `default_versions` object to the `manifest.yml` file. The `default_versions` object has two properties, `name` and `version`. For example:

```
default_versions:
- name: go
  version: 1.6.3
- name: other-dependency
  version: 1.1.1
```

To specify a default version:

1. Add the `default_version` object to your manifest, following the rules below. You can find a complete example manifest in the PAS go-buildpack repository.

2. Run the `default_version_for` script from the compile-extensions repository, passing the path of your `manifest.yml` and the dependency name as arguments. The following command uses the example manifest from step 1:

```
$ ./compile-extensions/bin/default_version_for manifest.yml go 1.6.3
```

Rules for Specifying a Default Version

The `buildpack-packager` script validates this object according to the following rules:

- You can create at most one entry under `default_versions` for a single dependency. The following example causes `buildpack-packager` to fail with an error because the manifest specifies two default versions for the same `go` dependency.

```
# Incorrect; will fail to package
default_versions:
- name: go
  version: 1.6.3
- name: go
  version: 1.7.5
```

- If you specify a `default_version` for a dependency, you must also list that dependency and version under the `dependencies` section of the manifest. The following example causes `buildpack-packager` to fail with an error because the manifest specifies `version: 1.9.2` for the `go` dependency, but lists `version: 1.7.5` under `dependencies`.

```
# Incorrect; will fail to package
default_versions:
- name: go
  version: 1.9.2

dependencies:
- name: go
  version: 1.7.5
  uri: https://storage.googleapis.com/golang/go1.7.5.linux-amd64.tar.gz
  md5: c8cb76e2308c792e2705c2eb1b55de95
  cf_stacks:
  - cflinuxfs2
```

Core Buildpack Communication Contract

This section describes the communication contract followed by the PAS core buildpacks. This contract enables buildpacks to interact with one another, so that developers can use multiple buildpacks with their applications.

Buildpack developers must ensure their custom buildpacks follow the contract.

This section uses the following placeholders:

- `IDX` is the zero-padded index matching the position of the buildpack in the priority list.
- `MD5` is the MD5 checksum of the buildpack's URL.

For all buildpacks that supply dependencies via `/bin/supply`:

- The buildpack must create `/tmp/deps/IDX/config.yml` to provide a name to subsequent buildpacks. This file may also contain miscellaneous configuration for subsequent buildpacks.

- The `config.yml` file should be formatted as follows, replacing `BUILDPACK` with the name of the buildpack providing dependencies and `YAML-OBJECT` with the YAML object that contains buildpack-specific configuration: `name: BUILDPACK config: YAML-OBJECT`
- The following directories may be created inside of /tmp/deps/IDX/ to provide dependencies to subsequent buildpacks:
  - `/bin` : Contains binaries intended for $PATH during staging and launch
  - `/lib` : Contains libraries intended for $LD_LIBRARY_PATH during staging and launch
  - `/include` : Contains header files intended for compilation during staging
  - `/pkgconfig` : Contains `pkgconfig` files intended for compilation during staging
  - `/env` : Contains environment vars intended for staging, loaded as `FILENAME=FILECONTENTS`
  - `/profile.d` : Contains scripts intended for `/app/.profile.d` , sourced before launch
- The buildpack may make use of previous non-final buildpacks by scanning `/tmp/deps/` for index-named directories containing `config.yml` .

For the last buildpack:

- To make use of dependencies provided by the previously applied buildpacks, the last buildpack must scan `/tmp/deps/` for index-named directories containing `config.yml.`
- To make use of dependencies provided by previous buildpacks, the last buildpack:
  - May use `/bin` during staging, or make it available in $PATH during launch
  - May use `/lib` during staging, or make it available in $LD_LIBRARY_PATH during launch
  - May use `/include` , `/pkgconfig` , or `/env` during staging
  - May copy files from `/profile.d` to `/tmp/app/.profile.d` during staging
  - May use the supplied config object in `config.yml` during the staging process

Deploy Apps with a Custom Buildpack

Once a custom buildpack has been created and pushed to a public git repository, the git URL can be passed via the cf CLI when pushing an app.

For example, for a buildpack that has been pushed to Github:

```
$ cf push my-new-app -b git://github.com/johndoe/my-buildpack.git
```

Alternatively, you can use a private git repository, with https and username/password authentication, as follows:

```
$ cf push my-new-app -b https://username:password@github.com/johndoe/my-buildpack.git
```

By default, PAS uses the default branch of the buildpack's git repository. You can specify a different branch using the git url as shown in the following example:

```
$ cf push my-new-app -b https://github.com/johndoe/my-buildpack.git#my-branch-name
```

Additionally, you can use tags in a git repository, as follows:

```
$ cf push my-new-app -b https://github.com/johndoe/my-buildpack#v1.4.2
```

The app will then be deployed to PAS, and the buildpack will be cloned from the repository and applied to the app.

> **Note**: If a buildpack is specified using `cf push -b` the `detect` step will be skipped and as a result, no buildpack `detect` scripts will be run.

Disable Custom Buildpacks

Operators can choose to disable custom buildpacks. For more information, see Disabling Custom Buildpacks.

> **Note**: A common development practice for custom buildpacks is to fork existing buildpacks and sync subsequent patches from upstream. To merge upstream patches to your custom buildpack, use the approach that Github recommends for syncing a fork.

Packaging Dependencies for Offline Buildpacks

Page last updated:

This topic describes the dependency storage options available to developers creating offline buildpacks.

About Offline Buildpacks

Online, or uncached, buildpacks require an Internet connection to download dependencies, such as language interpreters and compilers. Alternatively, you can create offline, or cached, buildpacks that are packaged with their dependencies. These offline buildpacks do not connect to the Internet when they are used to deploy Cloud Foundry apps.

> **Note**: Offline buildpacks may contain proprietary dependencies that require distribution licensing or export control measures.

You can find instructions for building the offline packages in the `README.md` file of each buildpack repository. For example, see the Java buildpack.

Package Dependencies in the Buildpack

The simplest way to package dependencies in a custom buildpack is to keep the dependencies in your buildpack source. However, this is strongly discouraged. Keeping the dependencies in your source consumes unnecessary space.

To avoid keeping the dependencies in source control, load the dependencies into your buildpack and provide a script for the operator to create a zipfile of the buildpack.

For example, the operator might complete the following process:

```
$ # Clones your buildpack
$ git clone http://YOUR-GITHUB-REPOSITORY.example.com/repository
$ cd SomeBuildPackName

$ # Creates a zipfile using your script
$ ./SomeScriptName
 ----> downloading-dependencies.... done
 ----> creating zipfile: ZippedBuildPackName.zip

$ # Adds the buildpack zipfile to the Cloud Foundry instance
$ cf create-buildpack SomeBuildPackName ZippedBuildPackName.zip 1
```

Pros

- Least complicated process for operators
- Least complicated maintenance process for buildpack developers

Cons

- Cloud Foundry admin buildpack uploads are limited to 1 GB, so the dependencies might not fit
- Security and functional patches to dependencies require updating the buildpack

Package Selected Dependencies in the Buildpack

This is a variant of the package dependencies in the buildpack method described above. In this variation, the administrator edits a configuration file such as `dependencies.yml` to include a limited subset of the buildpack dependencies, then packages and uploads the buildpack.

> **Note**: This approach is strongly discouraged. Please see the Cons section below for more information.

The administrator completes the following steps:

```
$ # Clones your buildpack
$ git clone http://YOUR-GITHUB-REPOSITORY.example.com/repository
$ cd SomeBuildPackName

$ # Selects dependencies
$ vi dependencies.yml # Or copy in a preferred config

$ # Builds a package using your script
$ ./package
 ----> downloading-dependencies.... done
 ----> creating zipfile: cobol_buildpack.zip

$ # Adds the buildpack to the Cloud Foundry instance
$ cf create-buildpack cobol-buildpack cobol_buildpack.zip 1

$ # Pushes an app using your buildpack
$ cd ~/my_app
$ cf push my-cobol-webapp -b cobol-buildpack
```

Pros

- Possible to avoid the Cloud Foundry admin buildpack upload size limit in one of two ways:
  - If the administrator chooses a limited subset of dependencies
  - If the administrator maintains different packages for different dependency sets

Cons

- More complex for buildpack maintainers
- Security updates to dependencies require updating the buildpack
- Proliferation of buildpacks that require maintenance:
  - For each configuration, there is an update required for each security patch
  - Culling orphan configurations may be difficult or impossible
  - Administrators need to track configurations and merge them with updates to the buildpack
  - May result in with a different config for each app

Rely on a Local Mirror

In this method, the administrator provides a compatible file store of dependencies. When running the buildpack, the administrator specifies the location of the file store. The buildpack should handle missing dependencies gracefully.

The administrator completes the following process:

```
$ # Clones your buildpack
$ git clone http://YOUR-GITHUB-REPOSITORY.example.com/repository
$ cd SomeBuildPackName

$ # Builds a package using your script
$ ./package https:///dependency/repository
 ----> creating zipfile: cobol_buildpack.zip

$ # Adds the buildpack to the Cloud Foundry instance
$ cf create-buildpack cobol-buildpack cobol_buildpack.zip 1

$ # Pushes an app using your buildpack
$ cd ~/my_app
$ cf push my-cobol-webapp -b cobol-buildpack
 ----> deploying app
 ----> downloading dependencies:
    https://OUR-INTERNAL-SITE.example.com/dependency/repository/dep1.tgz.... done
    https://OUR-INTERNAL-SITE.example.com/dependency/repository/dep2.tgz.... WARNING: dependency not found!
```

Pros

- Avoids the Cloud Foundry admin buildpack upload size limit
- Leaves the administrator completely in control of providing dependencies
- Security and functional patches for dependencies can be maintained separately on the mirror given the following conditions:
    - The buildpack is designed to use newer semantically versioned dependencies
    - Buildpack behavior does not change with the newer functional changes

Cons

- The administrator needs to set up and maintain a mirror
- The additional config option presents a maintenance burden

Merging from Upstream Buildpacks

Page last updated:

This topic describes how to maintain your forked buildpack by merging it with the upstream buildpack. This allows you to keep your fork updated with changes from the original buildpack, providing patches, updates, and new features.

The following procedure assumes that you are maintaining a custom buildpack that was forked from a Cloud Foundry system buildpack. However, you can use the same procedure to update a buildpack forked from any upstream buildpack.

To sync your forked buildpack with an upstream Cloud Foundry buildpack:

1. Navigate to your forked repository on GitHub and click **Compare** in the upper right to display the **Comparing changes** page. This page shows the unmerged commits between your forked buildpack and the upstream buildpack.

2. Inspect the unmerged commits and confirm that you want to merge them all.

3. In a terminal window, navigate to the forked repository and set the upstream remote as the Cloud Foundry buildpack repository.

```
$ cd ~/workspace/ruby-buildpack
$ git remote add upstream git@github.com:cloudfoundry/ruby-buildpack.git
```

4. Pull down the remote upstream changes.

```
$ git fetch upstream
```

5. Merge the upstream changes into the intended branch. You may need to resolve merge conflicts. This example shows merging the `master` branch of the upstream buildpack into the `master` branch of the forked buildpack.

```
$ git checkout master
$ git merge upstream/master
```

> **Note**: When merging upstream buildpacks, do not use `git rebase`. This approach is not sustainable because you confront the same merge conflicts repeatedly.

6. Run the buildpack test suite to ensure that the upstream changes do not break anything.

```
$ BUNDLE_GEMFILE=cf.Gemfile buildpack-build
```

7. Push the updated branch.

```
$ git push
```

Your forked buildpack is now synced with the upstream Cloud Foundry buildpack.

For more information about syncing forks, see the Github topic [Syncing a Fork](#).

Upgrading Dependency Versions

Page last updated:

This topic describes how to upgrade a dependency version in a custom buildpack. These procedures enable Cloud Foundry (CF) operators to maintain custom buildpacks that contain dependencies outside of the dependencies in the CF system buildpacks.

Cloud Foundry Buildpacks Team Process

Note: The procedures in this topic refer to the tools used by the CF buildpacks team, but they do not require the specific tools listed below. You can use any continuous integration (CI) system and workflow management tool to update dependencies in custom buildpacks.

The CF buildpacks team uses the following tools to update dependencies:

- A Concourse deployment of the buildpacks-ci pipelines
- The public-buildpacks-ci-robots GitHub repository
- Pivotal Tracker for workflow management

When the New Releases job in the notifications pipeline detects a new version of a tracked dependency in a buildpack, it creates a Tracker story about building and including the new version of the dependency in the buildpack manifests. It also posts a message as the `dependency-notifier` to the #buildpacks channel in the Cloud Foundry Slack channel.

Build the Binaries

For all dependencies, you must build the binary from source or acquire the binary as a tarball from a trusted source. For most dependencies, the CF buildpacks team builds the binaries from source.

Note: The steps below assume you are using a Concourse deployment of the `buildpacks-ci` pipelines and Pivotal Tracker.

To build the binary for a dependency, perform the following steps:

1. Navigate to the `public-buildpacks-ci-robots` directory and verify no uncommitted changes exist.

```
$ cd ~/workspace/public-buildpacks-ci-robots
$ git status
```

2. Run the `git pull` command in the directory to ensure it contains most recent version of the contents.

```
$ git pull -r
```

3. Navigate to the `binary-builds` directory.

```
$ cd binary-builds
```

4. Locate the YAML file for the buildpack you want to build a binary for. The directory contains YAML files for all the packages and dependencies tracked by the CF buildpacks team. Each YAML file correlates to the build queue for one dependency or package, and uses the naming format `DEPENDENCY-NAME.yml`. For example, the YAML file tracking the build queue for Ruby is named `ruby-builds.yml` and contains the following contents:

```
---
ruby: []
```

5. Different buildpacks use different signatures for verification. Determine which signature your buildpack requires by consulting the list in the buildpacks section of this topic and run follow the instructions to locate the SHA256, MD5, or GPG signature for the binary:
   - For the SHA256 of a file, run `shasum -a 256 FILE-NAME`.
   - For the MD5 of a file, run `md5 FILE-NAME`.
   - For the GPG signature (for Nginx), see the Nginx Downloads page.

6. Add the version and verification for the new binary to the YAML file as attributes of an element under the dependency name. For example, to build

the Ruby 2.3.0 binary verified with SHA256, add the following to the YAML file:

```
---
ruby:
- version: 2.3.0
  sha256: ba5ba60e5f1aa21b4ef8e9bf35b9ddb57286cb546aac4b5a28c71f459467e507
```

Note: Do not preface the version number with the name of the binary or language. For example, specify `2.3.0` for `version` instead of `ruby-2.3.0`.

You can enqueue builds for multiple versions at the same time. For example, to build both the Ruby 2.3.0 binary and the Ruby 2.3.1 binary, add the following to the YAML file:

```
---
ruby:
- version: 2.3.0
  sha256: ba5ba60e5f1aa21b4ef8e9bf35b9ddb57286cb546aac4b5a28c71f459467e507
- version: 2.3.1
  sha256: b87c738cb2032bf4920fef8e3864dc5cf8eae9d89d8d523ce0236945c5797dcd
```

7. Use the `git add` command to stage your changes:

```
$ git add .
```

8. Use the `git commit -m "YOUR-COMMIT-MESSAGE [#STORY-NUMBER]"` command to commit your changes using the Tracker story number. Replace `YOUR-COMMIT-MESSAGE` with your commit message, and `STORY-NUMBER` with the number of your Tracker story.

```
$ git commit -m "make that change [#1234567890]"
```

9. Run `git push` to push your changes to the remote origin.

```
$ git push
```

10. Pushing your changes triggers the binary building process, which you can monitor at the binary-builder pipeline of your own `buildpacks-ci` Concourse deployment. When the build completes, it adds a link to the Concourse build run to the Tracker story specified in the commit message for the new release.

Note: Binary builds are executed by the Cloud Foundry Binary Builder and the binary-builder pipeline.

Update Buildpack Manifests

After you build the binary for a dependency that you can access and download from a URL, follow the instructions below to add the dependency version to the buildpack manifest.

Note: The steps below assume you are using a Concourse deployment of the `buildpacks-ci` pipelines and Pivotal Tracker.

1. Navigate to the directory of the buildpack for which you want to update dependencies and run `git checkout develop` to check out the `develop` branch.

```
$ cd ~/workspace/ruby-buildpack
$ git checkout develop
```

2. Edit the `manifest.yml` file for the buildpack to add or remove dependencies.

```
dependencies:
 - name: ruby
   version: 2.3.0
   md5: 535342030a11abeb11497824bf642bf2
   uri: https://pivotal-buildpacks.s3.amazonaws.com/concourse-binaries/ruby/ruby-2.3.0-linux-x64.tgz
   cf_stacks:
     - cflinuxfs2
```

   - Follow the current structure of the manifest. For example, if the manifest includes the two most recent patch versions for each minor version of the language, you should also include the two most recent patch versions for each minor version of the language, such as both `ruby-2.1.9`

and `ruby-2.1.8`.
  - Copy the `uri` and the `md5` from the `build-BINARY-NAME` job that ran in the Concourse binary-builder pipeline and add them to the manifest.

  > **Note:** In the PHP buildpack, you may see a `modules` line for each PHP dependency in the manifest. Do not include this `modules` line in your new PHP dependency entry. The `modules` line will be added to the manifest by the `ensure-manifest-has-modules` Concourse job in the `php-buildpack` when you commit and push your changes. You can see this in the output logs of the `build-out` task.

3. Replace any other mentions of the old version number in the buildpack repository with the new version number. The CF buildpack team uses Ag for text searching.

```
$ ag OLD-VERSION
```

4. Run the following command to package and upload the buildpack, set up the org and space for tests in the specified CF deployment, and run the CF buildpack tests.

```
$ BUNDLE_GEMFILE=cf.Gemfile buildpack-build
```

   If the command fails, you may need to fix or change the tests, fixtures, or other parts of the buildpack.

5. Once the test suite completely passes, use git commands to stage, commit, and push your changes:

```
$ git add .
$ git commit -m "YOUR-MESSAGE[#TRACKER-STORY-ID]"
$ git push
```

6. Monitor the `LANGUAGE-buildpack` pipeline in Concourse. Once the test suite builds, the `specs-lts-develop` job and `specs-edge-develop` job, pass for the buildpack, you can deliver the Tracker story for the new Dependency release. Copy and paste links for the successful test suite builds into the Tracker story.

Buildpacks

The following list contains information about the buildpacks maintained by the CF buildpacks team.

Go Buildpack

Go:

- **Built from:** A tarred binary, `GO-VERSION.linux-amd64.tar.gz`, provided by Google on the Go Downloads page
- **Verified with:** The MD5 of the tarred binary
- **Example:** Using the Google Tarred Binary for Go 1.6.2

Godep:

- **Built from:** A source code `.tar.gz` file from the Godep Github releases page
- **Verified with:** The SHA256 of the source **Example:** Automated enqueuing of binary build for Godep 72

> **Note:** The buildpacks-ci binary-builder pipeline automates the process of detecting, uploading, and updating Godep in the manifest.

Node.js Buildpack

Node:

- **Verified with:** The SHA256 of the `node-vVERSION.tar.gz` file listed on `https://nodejs.org/dist/vVERSION/SHASUMS256.txt` For example, for Node version 4.4.6, the CF buildpacks team verifies with the SHA256 for `node-v4.4.6.tar.gz` on its SHASUMS256 page.
- **Example:** Enqueuing binary builds for Node 4.4.5 and 6.2.0

Python Buildpack

Python:

- **Verified with:** The MD5 of the `Gzipped source tarball`, listed on `https://www.python.org/downloads/release/python-VERSION/`, where `VERSION` has no periods. For example, for Python version `2.7.12`, use the MD5 for the `Gzipped source tarball` on its downloads page.
- **Example:** Enqueuing binary build for Python 2.7.12

Java Buildpack

OpenJDK:

- **Built from:** The tarred OpenJDK files managed by the CF Java Buildpack team
- **Verified with:** The MD5 of the tarred OpenJDK files

Ruby Buildpack

JRuby:

- **Verified with:** The MD5 of the Source `.tar.gz` file from the JRuby Downloads page
- **Example:** Enqueuing binary build for JRuby 9.1.2.0

Ruby:

- **Verified with:** The SHA256 of the source from the Ruby Downloads page
- **Example:** Enqueuing binary builds for Ruby 2.2.5 and 2.3.1

Bundler:

- **Verified with:** The SHA256 of the `.gem` file from Rubygems
- **Example:** Enqueuing binary build for Bundler 1.12.5

PHP Buildpack

PHP:

- **Verified with:** The SHA256 of the `.tar.gz` file from the PHP Downloads page.
- For PHP5 versions, the CF buildpacks team enqueues builds in the `php-builds.yml` file in the `binary-builds` branch. For PHP7 versions, the CF buildpacks team enqueues builds in the `php7-builds.yml` file in the `binary-builds` branch.
- **Example:** Enqueuing binary builds for PHP 5.5.37, 5.6.23, and 7.0.8

Nginx:

- **Verified with:** The `gpg-rsa-key-id` and `gpg-signature` of the version. The `gpg-rsa-key-id` is the same for each version/build, but the `gpg-signature` will be different. This information is located on the Nginx Downloads page.
- **Example:** Enqueuing binary build for Nginx 1.11.0

HTTPD:

- **Verified with:** The MD5 of the `.tar.bz2` file from the HTTPD Downloads page
- **Example:** Enqueuing binary build for HTTPD 2.4.20

Composer:

- **Verified with:** The SHA256 of the `composer.phar` file from the Composer Downloads page
- For Composer, there is no build process as the `composer.phar` file is the binary. In the manual process, connect to the `pivotal-buildpacks` S3 bucket using the correct AWS credentials. Create a new directory with the name of the composer version, for example `1.0.2`, and put the appropriate `composer.phar` file into that directory. For Composer `v1.0.2`, connect and create the `php/binaries/trusty/composer/1.0.2` directory. Then place the `composer.phar` file into that directory so the binary is available at `php/binaries/trusty/composer/1.0.2/composer.phar`.

> **Note**: The buildpacks-ci binary-builder pipeline automates the process of detecting, uploading, and updating Composer in the manifest.

- **Example**: Automated enqueuing of binary build for Composer 1.1.2

Staticfile Buildpack

Nginx:

- **Verified with**: The `gpg-rsa-key-id` and `gpg-signature` of the version. The `gpg-rsa-key-id` is the same for each version/build, but the `gpg-signature` will be different. This information is located on the Nginx Downloads page.
- **Example**: Enqueuing binary build for Nginx 1.11.0

Binary Buildpack

The Binary buildpack has no dependencies.

Using CI for Buildpacks

The Cloud Foundry (CF) Buildpacks team and other CF buildpack development teams use Concourse continuous integration (Concourse CI) pipelines to integrate new buildpack releases. This topic provides links to information that describes how to release new versions of Cloud Foundry buildpacks using Concourse CI, and how to update Ruby gems used for CF buildpack development.

Each of the following are applicable to all supported buildpack languages and frameworks:

- Releasing a New Buildpack Version
- Updating Buildpack-Related Gems

Releasing a New Buildpack Version

Page last updated:

This topic describes how to update and release a new version of a Cloud Foundry (CF) buildpack through the CF Buildpacks Team Concourse pipeline. Concourse is a continuous integration (CI) tool for software development teams. This is the process used by the CF Buildpacks Team and other CF buildpack development teams. You can use this process as a model for using Concourse to build and release new versions of your own buildpacks.

The Concourse pipelines for Cloud Foundry buildpacks are located in the `buildpacks-ci` Github repository.

Release a New Buildpack Version

To release a new buildpack version, perform the following:

1. Ensure you have downloaded the `buildpacks-ci` repository:

```
$ git clone https://github.com/cloudfoundry/buildpacks-ci.git
```

2. From the buildpack directory, check out the `develop` branch of the buildpack:

```
$ cd /system/path/to/buildpack
$ git checkout develop
```

3. Ensure you have the most current version of the repository:

```
$ git pull -r
```

4. Run `bump` to update the version in the buildpack repository:

```
$ /system/path/to/buildpacks-ci/scripts/bump
```

5. Modify the `CHANGELOG` file manually to condense recent commits into relevant changes. For more information, see Modify Changelogs.

6. Add and commit your changes:

```
$ git add VERSION CHANGELOG
$ git commit -m "Bump version to $(cat VERSION) [{insert story #}]"
```

7. Push your changes to the `develop` branch:

```
$ git push origin develop
```

Concourse Buildpack Workflow

If `buildpacks-ci` is not deployed to Concourse, manually add a Git tag to the buildpack and mark the tag as a release on Github.

If `buildpacks-ci` is deployed to Concourse, the buildpack update passes through the following life cycle:

1. Concourse triggers the `buildpack-to-master` job in the pipeline for the updated buildpack. This job merges develop onto the master branch of the buildpack.

2. The `detect-new-buildpack-and-upload-artifacts` job triggers in the pipeline for the updated buildpack. This job creates a cached and uncached buildpack and uploads them to an AWS S3 bucket.

3. The `specs-lts-master` and `specs-edge-master` jobs trigger and run the buildpack test suite and the buildpack-specific tests of the Buildpack Runtime Acceptance Tests (BRATS).

4. If you are using Pivotal Tracker, paste the links for the `specs-edge-master` and `specs-lts-master` builds in the related buildpack release story and deliver that story.

5. Your project manager can manually trigger the `buildpack-to-github` and `buildpack-to-pivnet` jobs on Concourse as part of the acceptance process. This releases the buildpack to Pivotal Network and to Github.

6. After the buildpack has been released to Github, the `cf-release` pipeline is triggered using the manual trigger of the `recreate-bosh-lite` job on that pipeline. If the new buildpack has been released to Github, the CF that is deployed for testing in the `cf-release` pipeline is tested against that new buildpack.

7. After the `cats` job has successfully completed, your project manager can ship the new buildpacks to the `cf-release` repository and create the new buildpack BOSH release by manually triggering the `ship-it` job.

Note: If errors occur during this workflow, you may need to remove unwanted tags. For more information, see Handle Unwanted Tags.

Modify Changelogs

The Ruby Buildpack changelog shows an example layout and content of a changelog. In general, changelogs follow these conventions:

- Reference public tracker stories whenever possible.
- Exclude unnecessary files
- Combine and condense commit statements into individual stories containing valuable changes.

Handle Unwanted Tags

If you encounter problems with the commit that contains the new version, change the target of the release tag by performing the following:

1. Ensure the repository is in a valid state and is building successfully.

2. Remove the tag from your local repository and from Github.

3. Start a build. The pipeline build script should re-tag the build if it is successful.

Updating Buildpack-Related Gems

Page last updated:

This topic describes how to update buildpack-packager and machete, used for CF system buildpack development.

`buildpack-packager` packages buildpacks and `machete` provides an integration test framework.

The CF Buildpacks team uses the gems-and-extensions pipeline to:

1. Run the integration tests for `buildpack-packager` and `machete`
2. Update the gems in the buildpacks managed by the team

Running the Update Process

Note: The steps below assume you are using a Concourse deployment of the `buildpacks-ci` pipelines

At the end of the process, there will be a new Github release and updates will be applied to the buildpacks.

To update the version of either gem in a buildpack:

1. Confirm that the test job `<gemname>-specs` for the gem to be updated successfully ran on the commit you plan to update.
2. Manually trigger the `<gemname>-tag` job to update ("bump") the version of the gem.
3. The `<gemname>-release` job will trigger. This will create a new Github release of the gem.
4. Each of the buildpack pipelines (e.g. the go-buildpack pipeline) has a job which watches for new releases of the gem. When a new release is detected, the buildpack's `cf.Gemfile` is updated to that release version.
5. The commit made to the buildpack's `cf.Gemfile` triggers the full integration test suite for that buildpack.

Note: The final step will trigger all buildpack test suites simultaneously, causing contention for available shared BOSH-lite test environments.

Services

Page last updated:

The documentation in this section is intended for developers and operators interested in creating Managed Services for Cloud Foundry. Managed Services are defined as having been integrated with Cloud Foundry via APIs, and enable end users to provision reserved resources and credentials on demand. For documentation targeted at end users, such as how to provision services and integrate them with applications, see Services Overview.

To develop Managed Services for Cloud Foundry, you'll need a Cloud Foundry instance to test your service broker with as you are developing it. You must have admin access to your CF instance to manage service brokers and the services marketplace catalog. For local development, we recommend using BOSH Lite to deploy your own local instance of Cloud Foundry.

Table of Contents

Pivotal

Overview

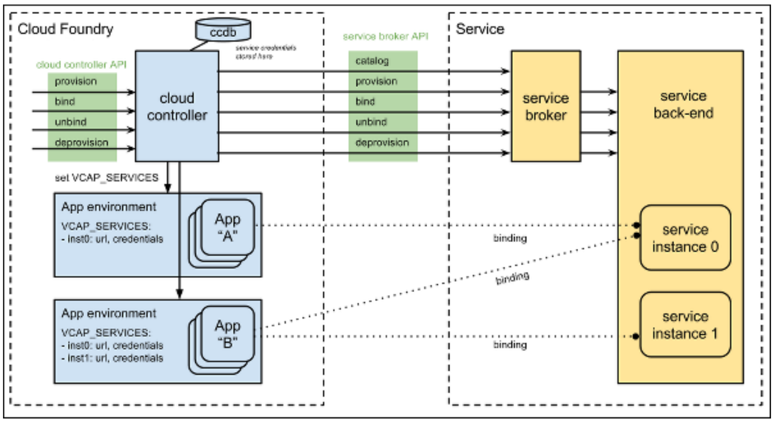
Page last updated:

Architecture & Terminology

Services are integrated with Cloud Foundry by implementing a documented API for which the cloud controller is the client; we call this the Service Broker API. This should not be confused with the cloud controller API, often used to refer to the version of Cloud Foundry itself; when one refers to “Cloud Foundry v2” they are referring to the version of the cloud controller API. The services API is versioned independently of the cloud controller API.

Service Broker is the term we use to refer to a component of the service which implements the service broker API. This component was formerly referred to as a Service Gateway, however as traffic between applications and services does not flow through the broker we found the term gateway caused confusion. Although gateway still appears in old code, we use the term broker in conversation, in new code, and in documentation.

Service brokers advertise a catalog of service offerings and service plans, as well as interpreting calls for provision (create), bind, unbind, and deprovision (delete). What a broker does with each call can vary between services; in general, ‘provision’ reserves resources on a service and 'bind' delivers information to an application necessary for accessing the resource. We call the reserved resource a Service Instance. What a service instance represents can vary by service; it could be a single database on a multi-tenant server, a dedicated cluster, or even just an account on a web application.



Implementation & Deployment

How a service is implemented is up to the service provider/developer. Cloud Foundry only requires that the service provider implement the service broker API. A broker can be implemented as a separate application, or by adding the required http endpoints to an existing service.

Because Cloud Foundry only requires that a service implements the broker API in order to be available to Cloud Foundry end users, many deployment models are possible. The following are examples of valid deployment models.

- Entire service packaged and deployed by BOSH alongside Cloud Foundry
- Broker packaged and deployed by BOSH alongside Cloud Foundry, rest of the service deployed and maintained by other means
- Broker (and optionally service) pushed as an application to Cloud Foundry user space
- Entire service, including broker, deployed and maintained outside of Cloud Foundry by other means

Managing Service Brokers

Page last updated:

This page assumes you are using cf CLI v6.16 or later.

In order to run many of the commands below, you must be authenticated with Cloud Foundry as an admin user or as a space developer.

Quick Start

Given a service broker that has implemented the Service Broker API, two steps are required to make its services available to end users in all orgs or a limited number of orgs by service plan.

1. Register a Broker
2. Make Plans Public

Register a Broker

Registering a broker causes Cloud Controller to fetch and validate the catalog from your broker, and save the catalog to the Cloud Controller database. The basic auth username and password which are provided when adding a broker are encrypted in Cloud Controller database, and used by the Cloud Controller to authenticate with the broker when making all API calls. Your service broker should validate the username and password sent in every request; otherwise, anyone could curl your broker to delete service instances. When the broker is registered with an URL having the scheme `https`, Cloud Controller will make all calls to the broker over HTTPS.

As of cf-release 229, CC API 2.47.0, Cloud Foundry supports two types of brokers:*standard brokers* and *space-scoped brokers*. A list of their differences follows:

Standard Brokers

- Publish service plans to specific orgs or all orgs in the deployment. Can also keep plans unavailable, or*private*.
- Created by admins, with the command `cf create-service-broker`

```
$ cf create-service-broker mybrokername someuser somethingsecure https://mybroker.example.com/
```

- Managed by admins
- Service plans are created private. Before anyone can use them, an admin must explicitly make them available within an org or across all orgs.

Space-Scoped Brokers

- Publish service plans only to users within the space they are created. Plans are unavailable outside of this space.
- Created by space developers using the command `cf create-service-broker` with the `--space-scoped` flag

```
$ cf create-service-broker mybrokername someuser somethingsecure https://mybroker.example.com/ --space-scoped
```

> **Note**: If a space developer runs `cf create-service-broker` without the `--space-scoped` flag, they receive an error.

- Managed by space developers
- Newly-created plans automatically publish to all users in their space.

Make Plans Public

After an admin creates a new service plan from a standard broker, no one can use it until the admin explicitly makes it available to users within a specific org or all orgs in the deployment.

See the Access Control topic for how to make standard broker service plans available to users.

Multiple Brokers, Services, Plans

Many service brokers may be added to a Cloud Foundry instance, each offering many services and plans. The following constraints should be kept in mind:

- It is not possible to have multiple brokers with the same name
- It is not possible to have multiple brokers with the same base URL
- The service ID and plan IDs of each service advertised by the broker must be unique across Cloud Foundry. GUIDs are recommended for these fields.

See Possible Errors below for error messages and what do to when you see them.

List Service Brokers

```
$ cf service-brokers
Getting service brokers as admin...Cloud Controller
OK

Name           URL
my-service-name https://mybroker.example.com
```

Update a Broker

Updating a broker is how to ingest changes a broker author has made into Cloud Foundry. Similar to adding a broker, update causes Cloud Controller to fetch the catalog from a broker, validate it, and update the Cloud Controller database with any changes found in the catalog.

Update also provides a means to change the basic auth credentials cloud controller uses to authenticate with a broker, as well as the base URL of the broker's API endpoints.

```
$ cf update-service-broker mybrokername someuser somethingsecure https://mybroker.example.com/
```

Rename a Broker

A service broker can be renamed with the `rename-service-broker` command. This name is used only by the Cloud Foundry operator to identify brokers, and has no relation to configuration of the broker itself.

```
$ cf rename-service-broker mybrokername mynewbrokername
```

Remove a Broker

Removing a service broker will remove all services and plans in the broker's catalog from the Cloud Foundry Marketplace.

```
$ cf delete-service-broker mybrokername
```

Note: Attempting to remove a service broker will fail if there are service instances for any service plan in its catalog. When planning to shut down or delete a broker, make sure to remove all service instances first. Failure to do so will leave orphaned service instances in the Cloud Foundry database. If a service broker has been shut down without first deleting service instances, you can remove the instances with the CLI; see Purge a Service.

Purge a Service

If a service broker has been shut down or removed without first deleting service instances from Cloud Foundry, you will be unable to remove the service broker or its services and plans from the Marketplace. In development environments, broker authors often destroy their broker deployments and need a way to clean up the Cloud Controller database.

The following command will delete a service offering, all of its plans, as well as all associated service instances and bindings from the Cloud Controller database, without making any API calls to a service broker. For services from v1 brokers, you must provide a provider with `-p PROVIDER`. Once all services for a broker have been purged, the broker can be removed normally.

```
$ cf purge-service-offering v1-test -p pivotal-software
Warning: This operation assumes that the service broker responsible for this
service offering is no longer available, and all service instances have been
deleted, leaving orphan records in Cloud Foundry's database. All knowledge of
the service will be removed from Cloud Foundry, including service instances and
service bindings. No attempt will be made to contact the service broker; running
this command without destroying the service broker will cause orphan service
instances. After running this command you may want to run either
delete-service-auth-token or delete-service-broker to complete the cleanup.

Really purge service offering v1-test from Cloud Foundry? y
OK
```

Purge a Service Instance

The following command will delete a single service instance, its service bindings and its service keys from the Cloud Controller database, without making any API calls to a service broker. This can be helpful in instances a Service Broker is not conforming to the Service Broker API and not returning a 200 or 410 to requests to delete the service instance.

```
$ cf purge-service-instance mysql-dev
WARNING: This operation assumes that the service broker responsible for this
service instance is no longer available or is not responding with a 200 or 410,
and the service instance has been deleted, leaving orphan records in Cloud
Foundry's database. All knowledge of the service instance will be removed from
Cloud Foundry, including service bindings and service keys.

Really purge service instance mysql-dev from Cloud Foundry?> y
Purging service mysql-dev...
OK
```

`purge-service-instance` requires cf-release v218 and cf CLI 6.14.0.

Catalog Validation Behaviors

When Cloud Foundry fetches a catalog from a broker, it will compare the broker's id for services and plans with the `unique_id` values for services and plans in the Cloud Controller database.

Event	Action
The catalog fails to load or validate.	Cloud Foundry will return a meaningful error that the broker could not be reached or the catalog was not valid.
A service or plan in the broker catalog has an ID that is not present among the `unique_id` values in the marketplace database.	A new record must be added to the marketplace database.
A service or plan in the marketplace database are found with a `unique_id` that matches the broker catalog's ID.	The marketplace must update the records to match the broker's catalog.
The database has plans that are not found in the broker catalog, and there are no associated service instances.	The marketplace must remove these plans from the database, and then delete services that do not have associated plans from the database.
The database has plans that are not found in the broker catalog, but there are provisioned instances.	The marketplace must mark the plan inactive and no longer display it or allow it to be provisioned.

Possible Errors

If incorrect basic auth credentials are provided:

```
Server error, status code: 500, error code: 10001, message: Authentication
failed for the service broker API.
Double-check that the username and password are correct:
  https://github-broker.a1-app.example.com/v2/catalog
```

If you receive the following errors, check your broker logs. You may have an internal error.

```
Server error, status code: 500, error code: 10001, message:
    The service broker response was not understood

Server error, status code: 500, error code: 10001, message:
    The service broker API returned an error from
    https://github-broker.a1-app.example.com/v2/catalog: 404 Not Found

Server error, status code: 500, error code: 10001, message:
    The service broker API returned an error from
    https://github-broker.primo.example.com/v2/catalog: 500 Internal Server Error
```

If your broker's catalog of services and plans violates validation of presence, uniqueness, and type, you will receive meaningful errors.

```
Server error, status code: 502, error code: 270012, message: Service broker catalog is invalid:
Service service-name-1
  service id must be unique
  service description is required
  service "bindable" field must be a boolean, but has value "true"
  Plan plan-name-1
    plan metadata must be a hash, but has value [{"bullets"=>["bullet1", "bullet2"]}]
```

Managing Access to Service Plans

Page last updated:

All new service plans from standard brokers are private by default. This means that when adding a new broker, or when adding a new plan to an existing broker's catalog, service plans won't immediately be available to end users. This lets an admin control which service plans are available to end users, and manage limited service availability.

Space-scoped brokers are registered to a specific space, and all users within that space can automatically access the broker's service plans. With space-scoped brokers, service visibility is not managed separately.

Prerequisites

- CLI v6.4.0
- Cloud Controller API v2.9.0 (cf-release v179)
- Admin user access; the following commands can be run only by an admin user

Display Access to Service Plans

The `service-access` CLI command enables an admin to see the current access control setting for every service plan in the marketplace, across all service brokers.

```
$ cf service-access
getting service access as admin...
broker: elasticsearch-broker
   service        plan     access    orgs
   elasticsearch  standard limited

broker: p-mysql
   service   plan       access   orgs
   p-mysql   100mb-dev   all
```

The `access` column shows values `all`, `limited`, or `none`, defined as follows:

- `all` - The service plan is available to all users, or *public*.
- `none` - No one can use the service plan; it is *private*.
- `limited` - The plan is available only to users within the orgs listed.

The `-b`, `-e`, and `-o` flags let you filter by broker, service, and org.

```
$ cf help service-access
NAME:
   service-access - List service access settings

USAGE:
   cf service-access [-b BROKER] [-e SERVICE] [-o ORG]

OPTIONS:
   -b     access for plans of a particular broker
   -e     access for plans of a particular service offering
   -o     plans accessible by a particular org
```

Enable Access to Service Plans

Admins use the `cf enable-service-access` command to give users access to service plans. The command grants access at the org level or across all orgs.

When an org has access to a plan, its users see the plan in the services marketplace (`cf marketplace`) and its Space Developer users can provision instances of the plan in their spaces.

Enable All-User Access to All Plans

Running `cf enable-service-access SERVICE-NAME` without any flags lets all users access every plan carried by the service. For example, the following command grants all-user access to all `p-mysql` service plans:

```
$ cf enable-service-access p-mysql
Enabling access to all plans of service p-mysql for all orgs as admin...
OK

$ cf service-access
getting service access as admin...
broker: p-mysql
  service  plan      access  orgs
  p-mysql  100mb-dev  all
```

Limit Access to Specific Orgs or Plans

The `-p` and `-o` flags to `cf enable-service-access` let the admin limit user access to specific service plans or orgs as follows:

- `-p PLAN` grants all users access to one service plan (access: `all`)
- `-o ORG` grants users in a specified org access to all plans (access: `limited`)
- `-p PLAN -o ORG` grants users in one org access to one plan (access: `limited`)

Run `cf help enable-service-access` to review these options from the command line.

Disable Access to Service Plans

Admins use the `cf disable-service-access` command to disable user access to service plans. The command denies access at the org level or across all orgs.

Disable Access to All Plans for All Users

Running `cf disable-service-access SERVICE-NAME` without any flags disables all user access to all plans carried by the service. For example, the following command denies any user access to all `p-mysql` service plans:

```
$ cf disable-service-access p-mysql
Disabling access to all plans of service p-mysql for all orgs as admin...
OK

$ cf service-access
getting service access as admin...
broker: p-mysql
  service   plan      access  orgs
  p-mysql   100mb-dev   none
```

Disable Access for Specific Orgs or Plans

The `-p` and `-o` flags to `cf disable-service-access` let the admin deny access to specific service plans or orgs as follows:

- `-p PLAN` disables user access to one service plan
- `-o ORG` disables access to all plans for users in a specified org
- `-p PLAN -o ORG` prevents users in one org from accessing one plan

Run `cf help disable-service-access` to review these options from the command line.

Limitations

- You cannot disable access to a service plan for an org if the plan is currently available to all orgs. You must first disable access for all orgs; then you can enable access for a particular org.

Dashboard Single Sign-On

Page last updated:

Introduction

Single Sign-On (SSO) enables Pivotal Cloud Foundry (PCF) users to authenticate with third-party service dashboards using their PCF credentials. Service dashboards are web interfaces which enable users to interact with some or all of the features the service offers. SSO provides a streamlined experience for users, limiting repeated logins and multiple accounts across their managed services. The user's credentials are never directly transmitted to the service since the OAuth protocol handles authentication.

Service Broker Responsibilities

Registering the Dashboard Client

1. A service broker must include the `dashboard_client` field in the JSON response from its catalog endpoint for each service implementing this feature. A valid response would appear as follows:

```
{
 "services": [
  {
   "id": "44b26033-1f54-4087-b7bc-da9652c2a539",
   ...
   "dashboard_client": {
    "id": "p-mysql-client",
    "secret": "p-mysql-secret",
    "redirect_uri": "http://p-mysql.example.com"
   }
  }
 ]
}
```

   The `dashboard_client` field is a hash containing three fields:
   - `id` is the unique identifier for the OAuth client that will be created for your service dashboard on the token server (UAA), and will be used by your dashboard to authenticate with the token server (UAA). If the client id is already taken, PCF will return an error when registering or updating the broker.
   - `secret` is the shared secret your dashboard will use to authenticate with the token server (UAA).
   - `redirect_uri` is used by the token server as an additional security precaution. UAA will not provide a token if the callback URL declared by the service dashboard doesn't match the domain name in `redirect_uri`. The token server matches on the domain name, so any paths will also match; e.g. a service dashboard requesting a token and declaring a callback URL of `http://p-mysql.example.com/manage/auth` would be approved if `redirect_uri` for its client is `http://p-mysql.example.com/`.

2. When a service broker which advertises the `dashboard_client` property for any of its services is added or updated, Cloud Controller will create or update UAA clients as necessary. This client will be used by the service dashboard to authenticate users.

Dashboard URL

A service broker should return a URL for the `dashboard_url` field in response to a provision request. Cloud Controller clients should expose this URL to users. `dashboard_url` can be found in the response from Cloud Controller to create a service instance, enumerate service instances, space summary, and other endpoints.

Users can then navigate to the service dashboard at the URL provided by `dashboard_url`, initiating the OAuth login flow.

Service Dashboard Responsibilities

OAuth Flow

When a user navigates to the URL from `dashboard_url`, the service dashboard should initiate the OAuth login flow. A summary of the flow can be found in section 1.2 of the OAuth RFC. OAuth expects the presence of an Authorization Endpoint and a Token Endpoint. In PCF, these endpoints are provided by the UAA. Clients can discover the location of UAA from Cloud Controller's info endpoint; in the response the location can be found in the `token_endpoint` field.

```
$ curl api.example.com/info
{"name":"vcap","build":"2222","support":"http://support.example.com","version
":2,"description":"Cloud Foundry sponsored by Pivotal","authorization_endpoint":
"https://login.example.com","token_endpoint":"https://uaa.example.com",
"allow_debug":true}
```

To enable service dashboards to support SSO for service instances created from different PCF instances, the /v2/info url is sent to service brokers in the `X-Api-Info-Location` header of every API call. A service dashboard should be able to discover this URL from the broker, and enabling the dashboard to contact the appropriate UAA for a particular service instance.

A service dashboard should implement the OAuth Authorization Code Grant type (UAA docs, RFC docs).

1. When a user visits the service dashboard at the value of `dashboard_url`, the dashboard should redirect the user's browser to the Authorization Endpoint and include its `client_id`, a `redirect_uri` (callback URL with domain matching the value of `dashboard_client.redirect_uri`), and list of requested scopes.
   Scopes are permissions included in the token a dashboard client will receive from UAA, and which Cloud Controller uses to enforce access. A client should request the minimum scopes it requires. The minimum scopes required for this workflow are `cloud_controller_service_permissions.read` and `openid`. For an explanation of the scopes available to dashboard clients, see On Scopes.

2. UAA authenticates the user by redirecting the user to the Login Server, where the user then approves or denies the scopes requested by the service dashboard. The user is presented with human readable descriptions for permissions representing each scope. After authentication, the user's browser is redirected back to the Authorization endpoint on UAA with an authentication cookie for the UAA.

3. Assuming the user grants access, UAA redirects the user's browser back to the value of `redirect_uri` the dashboard provided in its request to the Authorization Endpoint. The `Location` header in the response includes an authorization code.

   ```
   HTTP/1.1 302 Found
   Location: https://p-mysql.example.com/manage/auth?code=F45jH
   ```

4. The dashboard UI should then request an access token from the Token Endpoint by including the authorization code received in the previous step. When making the request the dashboard must authenticate with UAA by passing the client `id` and `secret` in a basic auth header. UAA will verify that the client id matches the client it issued the code to. The dashboard should also include the `redirect_uri` used to obtain the authorization code for verification.

5. UAA authenticates the dashboard client, validates the authorization code, and ensures that the redirect URI received matches the URI used to redirect the client when the authorization code was issues. If valid, UAA responds back with an access token and a refresh token.

Checking User Permissions

UAA is responsible for authenticating a user and providing the service with an access token with the requested permissions. However, after the user has been logged in, it is the responsibility of the service dashboard to verify that the user making the request to manage an instance currently has access to that service instance.

The service can accomplish this with a GET to the `/v2/service_instances/:guid/permissions` endpoint on the Cloud Controller. The request must include a token for an authenticated user and the service instance guid. The token is the same one obtained from the UAA in response to a request to the Token Endpoint, described above. .

Example Request:

```
curl -H 'Content-Type: application/json' \
    -H 'Authorization: bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjoid' \
    http://api.cloudfoundry.com/v2/service_instances/44b26033-1f54-4087-b7bc-da9652c2a539/permissions
```

Response:

```
{
  "manage": true,
  "read": true
}
```

The response includes the following fields which indicate the various user permissions for the given service instance: - `manage` – a `true` value indicates that the user has sufficient permissions to make changes to and update the service instance; `false` indicates insufficient permissions. - `read` – a `true` value indicates that the user has permission to access read-only diagnostic and monitoring information for the given service instance (e.g. permission to view a read-only dashboard); `false` indicates insufficient permissions.

Since administrators may change the permissions of users at any time, the service should check this endpoint whenever a user uses the SSO flow to access the service's UI.

On Scopes

Scopes let you specify exactly what type of access you need. Scopes limit access for OAuth tokens. They do not grant any additional permission beyond that which the user already has.

Minimum Scopes

The following two scopes are necessary to implement the integration. Most dashboard shouldn't need more permissions than these scopes enabled.

Scope	Permissions
`openid`	Allows access to basic data about the user, such as email addresses
`cloud_controller_service_permissions.read`	Allows access to the CC endpoint that specifies whether the user can manage a given service instance

Additional Scopes

Dashboards with extended capabilities may need to request these additional scopes:

Scope	Permissions
`cloud_controller.read`	Allows read access to all resources the user is authorized to read
`cloud_controller.write`	Allows write access to all resources the user is authorized to update / create / delete

Reference Implementation

The MySQL Service Broker is an example of a broker that also implements a SSO dashboard. The login flow is implemented using the OmniAuth library and a custom UAA OmniAuth Strategy. See this OmniAuth wiki page for instructions on how to create your own strategy.

The UAA OmniAuth strategy is used to first get an authorization code, as documented in this section of the UAA documentation. The user is redirected back to the service (as specified by the `callback_path` option or the default `auth/cloudfoundry/callback` path) with the authorization code. Before the application / action is dispatched, the OmniAuth strategy uses the authorization code to get a token and uses the token to request information from UAA to fill the `omniauth.auth` environment variable. When OmniAuth returns control to the application, the `omniauth.auth` environment variable hash will be filled with the token and user information obtained from UAA as seen in the Auth Controller.

Restrictions

- UAA clients are scoped to services. There must be a `dashboard_client` entry for each service that uses SSO integration.
- Each `dashboard_client id` must be unique across the CloudFoundry deployment.

Resources

- OAuth

- Example broker with SSO implementation
- Cloud Controller API Docs
- User Account and Authentication (UAA) Service APIs

Example Service Brokers

Page last updated:

The following example service broker applications have been developed - these are a great starting point if you are developing your own service broker.

Ruby

- GitHub Repository service - this is designed to be an easy-to-read example of a service broker, with complete documentation, and comes with a demo app that uses the service. The broker can be deployed as an application to any Cloud Foundry instance or hosted elsewhere. The service broker uses GitHub as the service back end.
- MySQL database service - this broker and its accompanying MySQL server are designed to be deployed together as a BOSH release. BOSH is used to deploy or upgrade the release, monitors the health of running components, and restarts or recreates unhealthy VMs. The broker code alone can be found here.

Java

- Spring Cloud - Cloud Foundry Service Broker - This implements the REST contract for service brokers and the artifacts are published to the Spring Maven repository. This greatly simplifies development: include a single dependency in Gradle, implement interfaces, and configure. A sample implementation has been provided for MongoDB.
- MySQL Java Broker - a Java port of the Ruby-based MySQL broker above.

Go

- Asynchronous Service Broker for AWS EC2 - This broker implements support for the experimental Asynchronous Service Operations, and calls AWS APIs to provision EC2 VMs.

Binding Credentials

Page last updated:

A bindable service returns credentials that an application can consume in response to the `cf bind` API call. Cloud Foundry writes these credentials to the `VCAP_SERVICES` environment variable. In some cases, buildpacks write a subset of these credentials to other environment variables that frameworks might need.

Choose from the following list of credential fields if possible, though you can provide additional fields as needed. Refer to the Using Bound Services section of the *Managing Service Instances with the CLI* topic for information about how these credentials are consumed.

Note: If you provide a service that supports a connection string, provide the `uri` key for buildpacks and application libraries to use.

CREDENTIALS	DESCRIPTION
uri	Connection string of the form `DB-TYPE://USERNAME:PASSWORD@HOSTNAME:PORT/NAME`, where `DB-TYPE` is a type of database such as mysql, postgres, mongodb, or amqp.
hostname	FQDN of the server host
port	Port of the server host
name	Name of the service instance
vhost	Name of the messaging server virtual host - a replacement for a `name` specific to AMQP providers
username	Server user
password	Server password

The following is an example output of `ENV['VCAP_SERVICES']`.

Note: Depending on the types of databases you are using, each database might return different credentials.

```
VCAP_SERVICES=
{
  cleardb: [
    {
      name: "cleardb-1",
      label: "cleardb",
      plan: "spark",
      credentials: {
        name: "ad_c6f4446532610ab",
        hostname: "us-cdbr-east-03.cleardb.com",
        port: "3306",
        username: "b5d435f40dd2b2",
        password: "ebfc00ac",
        uri: "mysql://b5d435f40dd2b2:ebfc00ac@us-cdbr-east-03.cleardb.com:3306/ad_c6f4446532610ab",
        jdbcUrl: "jdbc:mysql://b5d435f40dd2b2:ebfc00ac@us-cdbr-east-03.cleardb.com:3306/ad_c6f4446532610ab"
      }
    }
  ],
  cloudamqp: [
    {
      name: "cloudamqp-6",
      label: "cloudamqp",
      plan: "lemur",
      credentials: {
        uri: "amqp://ksvyjmiv:IwN6dCdZmeQD4O0ZPKpu1YOaLx1he8wo@lemur.cloudamqp.com/ksvyjmiv"
      }
    }
    {
      name: "cloudamqp-9dbc6",
      label: "cloudamqp",
      plan: "lemur",
      credentials: {
        uri: "amqp://vhuklnxa:9lNFxpTuJsAdTts98vQIdKHW3MojyMyV@lemur.cloudamqp.com/vhuklnxa"
      }
    }
  ],
  rediscloud: [
    {
      name: "rediscloud-1",
      label: "rediscloud",
      plan: "20mb",
      credentials: {
        port: "6379",
```

```
        host: "pub-redis-6379.us-east-1-2.3.ec2.redislabs.com",
        password: "1M5zd3QfWi9nUyya"
      }
    },
  ],
}
```

Application Log Streaming

Page last updated:

By binding an application to an instance of an applicable service, Cloud Foundry will stream logs for the bound application to the service instance.

- Logs for all apps bound to a log-consuming service instance will be streamed to that instance
- Logs for an app bound to multiple log-consuming service instances will be streamed to all instances

To enable this functionality, a service broker must implement the following:

1. In the catalog endpoint, the broker must include `requires: syslog_drain`. This minor security measure validates that a service returning a `syslog_drain_url` in response to the bind operation has also declared that it expects log streaming. If the broker does not include `requires: syslog_drain`, and the bind request returns a value for `syslog_drain_url`, Cloud Foundry will return an error for the bind operation.

2. In response to a bind request, the broker should return a value for `syslog_drain_url`. The syslog URL has a scheme of syslog, syslog-tls, or https and can include a port number. For example:
   `"syslog_drain_url": "syslog://logs.example.com:1234"`

How does it work?

1. Service broker returns a value for `syslog_drain_url` in response to bind

2. Loggregator periodically polls CC `/v2/syslog_drain_urls` for updates

3. Upon discovering a new `syslog_drain_url`, Loggregator identifies the associated app

4. Loggregator streams app logs for that app to the locations specified by the service instances' `syslog_drain_url`s

Users can manually configure app logs to be streamed to a location of their choice using User-provided Service Instances. For details, see Using Third-Party Log Management Services.

Route Services

Page last updated:

This documentation is intended for service authors who are interested in offering a service to a Cloud Foundry (CF) services marketplace. Developers interested in consuming these services can read the Manage Application Requests with Route Services topic.

Introduction

Cloud Foundry application developers may wish to apply transformation or processing to requests before they reach an application. Common examples of use cases include authentication, rate limiting, and caching services. Route Services are a kind of Marketplace Service that developers can use to apply various transformations to application requests by binding an application's route to a service instance. Through integrations with service brokers and, optionally, with the Cloud Foundry routing tier, providers can offer these services to developers with a familiar, automated, self-service, and on-demand user experience.

> **Note**: The procedures in this topic use the Cloud Foundry Command Line Interface (cf CLI). You can also manage route services using Apps Manager. For more information, see the Manage Route Services section of the *Managing Apps and Service Instances Using Apps Manager* topic.

Architecture

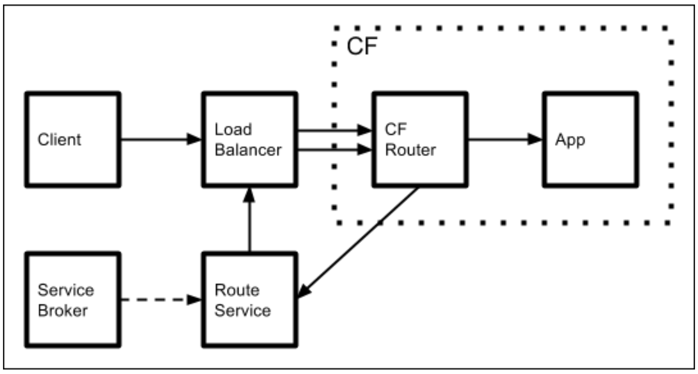
Cloud Foundry supports the following three models for Route Services:

- Fully-brokered services
- Static, brokered services
- User-provided services

In each model, you configure a route service to process traffic addressed to an app.

Fully-Brokered Service

In the fully-Brokered Service model, the CF router receives all traffic to apps in the deployment before any processing by the route service. Developers can bind a route service to any app, and if an app is bound to a route service, the CF router sends its traffic to the service. After the route service processes requests, it sends them back to the load balancer in front of the CF router. The second time through, the CF router recognizes that the route service has already handled them, and forwards them directly to app instances.



The route service can run inside or outside of CF, so long as it fulfills the Service Instance Responsibilities to integrate it with the CF router. A service broker publishes the route service to the CF marketplace, making it available to developers. Developers can then create an instance of the service and bind it to their apps with the following commands:

`cf create-service BROKER-SERVICE-PLAN SERVICE-INSTANCE` `cf bind-route-service YOUR-APP-DOMAIN SERVICE-INSTANCE [--hostname HOSTNAME] [--path PATH]`

Developers configure the service either through the service provider's web interface or by passing arbitrary parameters to their `cf create-service` call through the `-c` flag.

Advantages:

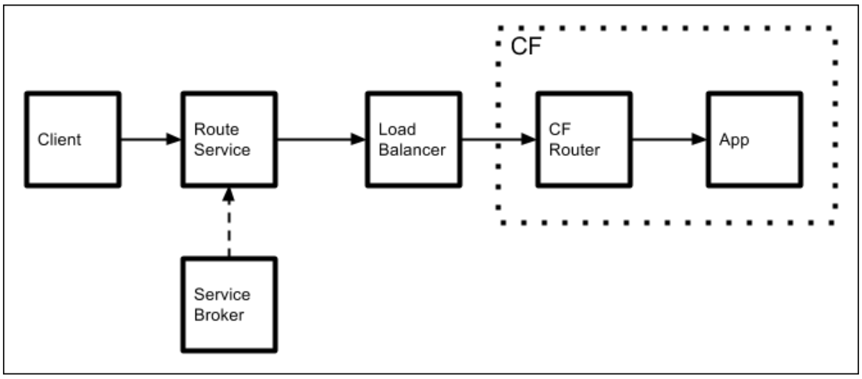
- Developers can use a Service Broker to dynamically configure how the route service processes traffic to specific applications.
- Adding route services requires no manual infrastructure configuration.
- Traffic to apps that do not use the service makes fewer network hops because requests for those apps do not pass through the route service.

Disadvantages:

- Traffic to apps that use the route service makes additional network hops, as compared to the static model.

Static, Brokered Service

In the static, brokered service model, an operator installs a static routing service, which might be a piece of hardware, in front of the load balancer. The routing service runs outside of Cloud Foundry and receives traffic to all apps running in the CF deployment. The service provider creates a service broker to publish the service to the CF marketplace. As with a fully-brokered service, a developer can use the service by instantiating it with `cf create-service` and binding it to an app with `cf bind-route-service`.



In this model, you configure route services on an app-by-app basis. When you bind a service to an app, the service broker directs the routing service to process that app's traffic rather than pass the requests through unchanged.

Advantages:

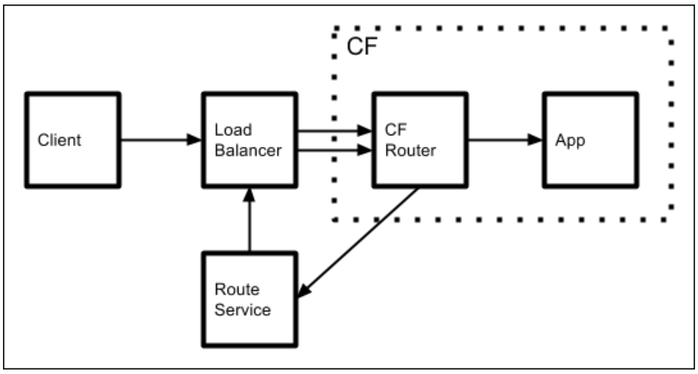
- Developers can use a Service Broker to dynamically configure how the route service processes traffic to specific applications.
- Traffic to apps that use the route service takes fewer network hops.

Disadvantages:

- Adding route services requires manual infrastructure configuration.
- Traffic to apps that do not use the route service make unnecessary network hops. Requests for all apps hosted by the deployment pass through the route service component.

User-Provided Service

If a route service is not listed in the CF marketplace by a broker, a developer can still bind it to their app as a user-provided service. The service can run anywhere, either inside or outside of CF, but it must fulfill the integration requirements described in Service Instance Responsibilities. The service also needs to be reachable by an outbound connection from the CF router.



This model is identical to the fully-brokered service model, except without the broker. Developers configure the service manually, outside of Cloud Foundry. They can then create a user-provided service instance and bind it to their application with the following commands, supplying the URL of their route service:

```
cf create-user-provided-service SERVICE-INSTANCE -r ROUTE-SERVICE-URL
```

```
cf bind-route-service DOMAIN SERVICE-INSTANCE [--hostname HOSTNAME]
```

Advantages:

- Adding route services requires no manual infrastructure configuration.
- Traffic to apps that do not use the service makes fewer network hops because requests for those apps do not pass through the route service.

Disadvantages:

- Developers must manually provision and configure route services out of the context of Cloud Foundry because no service broker automates these operations.
- Traffic to apps that use the route service makes additional network hops, as compared to the static model.

Architecture Comparison

The models above require the broker and service instance responsibilities summarized in the following table:

Route Services Architecture	Fulfills CF Service Instance Responsibilities	Fulfills CF Broker Responsibilities
Fully-Brokered	Yes	Yes
Static Brokered	No	Yes
User-Provided	Yes	No

Enabling Route Services in Pivotal Cloud Foundry

You configure Route Services for your deployment in the PAS tile, under **Settings** > **Networking**. Depending on your infrastructure, refer to the PAS configuration topics for AWS, OpenStack, or vSphere.

Service Instance Responsibilities

The following applies only when a broker returns `route_service_url` in the bind response.

How It Works

Binding a service instance to a route associates the `route_service_url` with the route in the CF router. All requests for the route are proxied to the URL specified by `route_service_url`.

Once a route service completes its function it may choose to forward the request to the originally requested URL or to another location, or it may choose to reject the request; rejected requests will be returned to the originating requestor. The CF router includes a header that provides the originally requested URL, as well as two headers that are used by the router itself to validate the request sent by the route service. These headers are described below in

Headers.

Headers

The following HTTP headers are added by the Gorouter to requests forwarded to route services.

X-CF-Forwarded-Url

The `X-CF-Forwarded-Url` header contains the originally requested URL. The route service may choose to forward the request to this URL or to another.

X-CF-Proxy-Signature

The `X-CF-Proxy-Signature` header contains an encypted value which only the Gorouter can decrypt.

The header contains the originally requested URL and a timestamp. When this header is present, the Gorouter will reject the request if the requested URL does not match that in the header, or if a timeout has expired.

`X-CF-Proxy-Signature` also signals to the Gorouter that a request has transited a route service. If this header is present, the Gorouter will not forward the request to a route service, preventing recursive loops. For this reason, route services should not strip off the `X-CF-Proxy-Signature` and `X-CF-Proxy-Metadata` headers.

If the route service forwards the request to a URL different from the originally requested one, and the URL resolves to a route for an application on Cloud Foundry, the route must not have a bound route service or the request will be rejected, as the requested URL will not match the one in the `X-CF-Proxy-Signature` header.

X-CF-Proxy-Metadata

The `X-CF-Proxy-Metadata` header aids in the encryption and description of `X-CF-Proxy-Signature`.

SSL Certificates

When Cloud Foundry is deployed in a development environment, certificates hosted by the load balancer are self-signed, and not signed by a trusted certificate authority. When the route service finishes processing an inbound request and makes a call to the value of `X-CF-Forwarded-Url`, be prepared to accept the self-signed certificate when integrating with a non-production deployment of Cloud Foundry.

Timeouts

Route services must forward the request to the application route within 60 seconds.

In addition, all requests must respond in 900 seconds.

Broker Responsibilities

Catalog Endpoint

Brokers must include `requires: ["route_forwarding"]` for a service in the catalog endpoint. If this is not present, Cloud Foundry will not permit users to bind an instance of the service to a route.

Binding Endpoint

When users bind a route to a service instance, Cloud Foundry sends a bind request to the broker, including the route address with `bind_resource.route`. A route is an address used by clients to reach apps mapped to the route. The broker may return `route_service_url`, containing a URL where Cloud Foundry should proxy requests for the route. This URL must have an `https` scheme, otherwise the Cloud Controller rejects the binding. `route_service_url` is optional, and not returning this field enables a broker to dynamically configure a network component already in the request path for the route, requiring no change in the CF router.

Example Route Services

- Logging Route Service: This route service can be pushed as an app to Cloud Foundry. It fulfills the service instance responsibilities above and logs requests received and sent. It can be used to see the route service integration in action by tailing its logs.
- Rate Limiting Route Service: This example route service is a simple Cloud Foundry app that provides rate limiting to control the rate of traffic to an application.
- Spring Boot Example: Logs requests received and sent; written in Spring Boot

Tutorial

The following instructions show how to use the Logging Route Service described in Example Route Services to verify that when a route service is bound to a route, requests for that route are proxied to the route service.

A video of this tutorial is available on Youtube.

These commands requires the Cloud Foundry Command Line Interface (cf CLI) version 6.16 or later.

1. Push the Logging Route Service as an app.

```
$ cf push logger
```

2. Create a user-provided service instance, and include the route of the Logging Route Service you pushed as `route_service_url`. Be sure to use `https` for the scheme.

```
$ cf create-user-provided-service mylogger -r https://logger.cf.example.com
```

3. Push a sample app like Spring Music. By default this creates a route `spring-music.cf.example.com`.

```
$ cf push spring-music
```

4. Bind the user-provided service instance to the route of your sample app. The `bind-route-service` command takes a route and a service instance; the route is specified in the following example by domain `cf.example.com` and hostname `spring-music`.

```
$ cf bind-route-service cf.example.com mylogger --hostname spring-music
```

5. Tail the logs for your route service.

```
$ cf logs logger
```

6. Send a request to the sample app and view in the route service logs that the request is forwarded to it.

```
$ curl spring-music.cf.example.com
```

Supporting Multiple Cloud Foundry Instances

Page last updated:

It is possible to register a service broker with multiple Cloud Foundry instances. It may be necessary for the broker to know which Cloud Foundry instance is making a given request. For example, when using Dashboard Single Sign-On, the broker is expected to interact with the authorization and token endpoints for a given Cloud Foundry instance.

There are two strategies that can be used to discover which Cloud Foundry instance is making a given request.

Routing & Authentication

The broker can use unique credentials and/or a unique url for each Cloud Foundry instance. When registering the broker, different Cloud Foundry instances can be configured to use different base urls that include a unique id. For example:

- On Cloud Foundry instance 1, the service broker is registered with the url `broker.example.com/123`
- On Cloud Foundry instance 2, the service broker is registered with the url `broker.example.com/456`

X-Api-Info-Location Header

All calls to the broker from Cloud Foundry include an `X-Api-Info-Location` header containing the `/v2/info` url for that instance. The `/v2/info` endpoint will return further information, including the location of that Cloud Foundry instance's UAA.

Support for this header was introduced in cf-release v212.

Pivotal

Loggregator Logs and Metrics

This documentation describes the Loggregator system, which aggregates and streams logs and metrics from all of the user apps and system components in Pivotal Application Service.

Table of Contents

Overview of the Loggregator System

Page last updated:

Loggregator gathers and streams logs and metrics from user apps in a Pivotal Cloud Foundry (PCF) deployment as well as metrics from PCF components. See the Loggregator repository on GitHub.

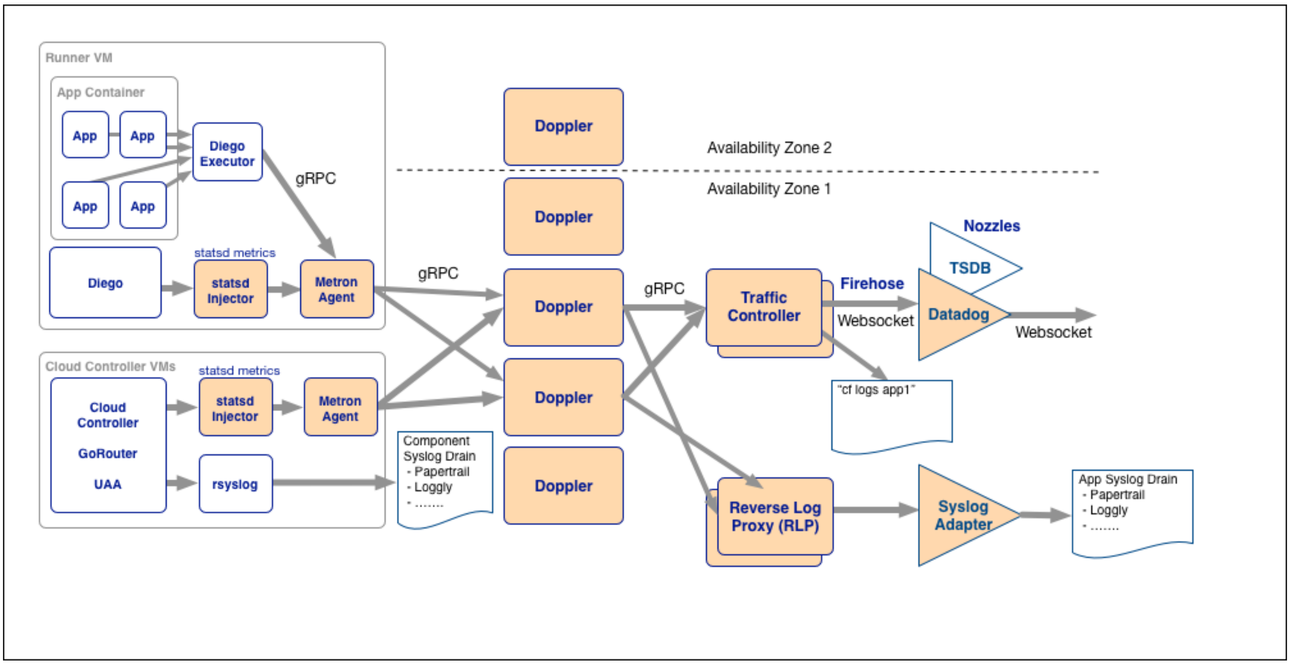
Using Loggregator

The primary use cases for Loggregator include the following:

- App developers can tail their application logs or dump the recent logs from the Cloud Foundry Command Line Interface (cf CLI), or stream these to a third-party log archive and analysis service.
- Operators and administrators can access the **Loggregator** Firehose, the combined stream of logs from all apps, and the metrics data from Cloud Foundry components.
- Operators can deploy **nozzles** to the Firehose. A nozzle is a component that monitors the Firehose for specified events and metrics, and streams this data to external services.

Loggregator Architecture

The diagram below shows the architecture of Loggregator, including the Cloud Foundry components that it interacts with.



To see a larger version of this diagram, click here.

Note: The Loggregator system uses gRPC for communication between the Metron Agent and the Doppler, and between the Doppler and the Traffic Controller. This improves the stability and the performance of the Loggregator system, but it may require operators to scale their Dopplers.

Source

Sources are logging agents that run on the Cloud Foundry components.

Metron

Metron Agents are colocated with sources. They collect logs and forward them to the Doppler servers.

Doppler

Dopplers gather logs from the Metron Agents, store them in temporary buffers, and forward them to the Traffic Controller or to third-party syslog drains.

Traffic Controller

The Traffic Controller handles client requests for logs. It gathers and collates messages from all Doppler servers, and provides external API and message translation as needed for legacy APIs. The Traffic Controller also exposes the Firehose.

Firehose

The Firehose is a WebSocket endpoint that streams all the event data coming from a Cloud Foundry deployment. The data stream includes logs, HTTP events, and container metrics from all applications, and metrics from all Cloud Foundry system components. Logs from system components such as the Cloud Controller are not included in the Firehose and are typically accessed through rsyslog configuration.

Because the data coming from the Firehose may contain sensitive information, such as customer information in the application logs, only users with the correct permissions can access the Firehose.

The Traffic Controller serves the Firehose over WebSocket at the `/firehose` endpoint. The events coming out of the Firehose are formatted as protobuf messages conforming to the dropsonde protocol.

You can discover the address of the Traffic Controller by hitting the `info` endpoint on the API and retrieving the value of the `doppler_logging_endpoint`.

Example for a BOSH Lite CF environment:

```
$ cf curl /v2/info | jq .doppler_logging_endpoint
wss://doppler.192.0.2.34.xip.io:443
```

Difference Between Logs and Metrics

The Firehose carries both logs and metrics, which differ as follows:

- **Metrics**
  - Report a measurement from a VM, or the status of a VM, at its timestamp time
  - Follow dropsonde or statsd protocol
  - Can route to a monitoring platform to trigger alerts
- **Logs**
  - Report events detected, actions taken, errors, or any other messages the operator or developer wanted to generate
  - Follow the syslog standard
  - Are not used to trigger alerts

CF Syslog Drain

This section describes the CF Syslog Drain components of Loggregator.

- **What CF Syslog Drain does**:
  - Loggregator uses the CF Syslog Drain Release to support developers who want to stream app logs to a syslog-compatible aggregation or analytics service. See Streaming Application Logs to Log Management Services.
- **What CF Syslog Drain consists of**:
  - CF Syslog Drain includes the Reverse Log Proxy (RLP) and Syslog Adapter components described below. These components run on VMs deployed

with Pivotal Application Service (PAS) that you can scale independently to support large numbers of user-provided syslog drains.

- **How you can scale**:
  - See Configuring System Logging in PAS.
- **When to scale**:
  - For guidance on scaling, see the CF Syslog Drain Performance Scaling Indicators section of the *Monitoring Pivotal Cloud Foundry* guide.

Reverse Log Proxy (RLP)

RLPs are a BOSH jobs colocated with the Traffic Controller that collect logs from Dopplers and forward them to Syslog Adapters. You can scale this component based on your overall log volume.

Syslog Adapter

Syslog Adapters are BOSH VMs that manage connections with and write to syslog services, or drains. You can scale this component based on the number of drains. For more information about Syslog Adapter capacity planning, see Scaling Loggregator.

Related Components

This section provides information about the components that are related to the Loggregator system.

BOSH System Metrics Forwarder

PCF uses the following components to send BOSH-reported system metrics to Loggregator:

- The **BOSH System Metrics Plugin** is deployed on the BOSH Director. This plugin reads health events such as VM heartbeats and alerts from the BOSH Health Monitor JSON plugin and streams them to the BOSH System Metrics Server.
- The **BOSH System Metrics Server** is deployed on the BOSH Director. The server accepts connections from the BOSH System Metrics Forwarder and streams the health events to it over gRPC as follows:
  - If two clients connect to the BOSH System Metrics Server using the same subscription ID, the server evenly distributes the event stream between them.
  - If two clients connect to the BOSH System Metrics Server using different subscription IDs, each client receives a copy of the event stream.
- The **BOSH System Metrics Forwarder** is colocated on the Traffic Controller. It initiates connections to the BOSH System Metrics Server and receives alerts and heartbeats over secure gRPC. The BOSH System Metrics Forwarder sends heartbeat events as envelopes to Loggregator through a colocated Metron Agent. It does not forward alerts.

Nozzles

Nozzles are programs which consume data from the Loggregator Firehose. Nozzles can be configured to select, buffer, and transform data, and forward it to other applications and services. Example nozzles include the following:

- The JMX Bridge OpenTSDB Firehose Nozzle, which installs with JMX Bridge
- The Datadog nozzle, which publishes metrics coming from the Firehose to Datadog
- The Syslog nozzle, which filters out log messages coming from the Firehose and sends it to a syslog server

For more information about nozzles, see the Nozzle Tutorial.

Loggregator Guide for Cloud Foundry Operators

Page last updated:

This topic contains information for Cloud Foundry deployments operators about how to configure the Loggregator system to avoid data loss with high volumes of logging and metrics data.

Loggregator Message Throughput and Reliability

For determining the message throughput and reliability rates of your Loggregator system, see the section below.

Measuring Message Throughput

To measure the message throughput of the Loggregator system, you can monitor the total number of egress messages from all Metrons on your platform using the `metron.egress` metric.

If you do not use a monitoring platform, you can follow the instructions below to measure the overall message throughput of your Loggregator system:

1. Log in to the Cloud Foundry Command Line Interface (cf CLI) with your admin credentials:

```
$ cf login
```

2. Install the Cloud Foundry Firehose plugin.

3. Install Pipe Viewer:

```
$ apt-get install pv
```

4. Run the following command:

```
$ cf nozzle -n | pv -l -i 10 -r > /dev/null
```

Measuring Message Reliability

To measure the message reliability rate of your Loggregator system, you can run black-box tests. If you want to use this method, see the open-source cf-logmon app and the configuration instructions provided in the README.md file.

Scaling Loggregator

Most Loggregator configurations are set to use preferred resource defaults. If you want to customize these defaults or plan the capacity of your Loggregator system, see the formulas below.

Doppler

Doppler resources can require scaling to accommodate your overall log and metric volume. Pivotal Application Service recommends the following formula for determining the number of Doppler instances you need to achieve a loss rate of < 1%:

Number of Doppler instances = Number of logs per second / 2,000

Because it can be challenging to understand the ratio of metrics to logs, Pivotal Application Service also recommends monitoring and scaling Doppler based on its ingress traffic. To do this, you need to sum two metrics and rate them per second:

Number of Doppler instances = `doppler.ingress` + `DopplerServer.listeners.receivedEnvelopes` */ 10,000*

Using maximum values over a two-week period is a recommended approach for ingress-based capacity planning.

Traffic Controller

Traffic Controller resources are usually scaled in line with Doppler resources. Pivotal Application Service recommends the following formula for determining the number of Traffic Controller instances:

Number of Traffic Controller instances = Number of Doppler instances / 4

In addition, Traffic Controller resources can require scaling to accommodate the number of your log streams and Firehose subscriptions.

Syslog Adapter

Syslog Adapter is a Loggregator component that manages user-provided syslog drains. This component should be scaled depending on the number of your drain bindings.

Note: A drain binding is a syslog destination associated with an app. Apps can have multiple bindings.

Pivotal Application Service (PAS) recommends the following formula for determining the number of Syslog Adapter instances:

Number of Syslog Adapter instances = Number of drain bindings / 500

You can use the `scheduler.drains` and `scheduler.adapters` metrics to configure auto-scaling of Syslog Adapters.

See Configuring Logging in PAS for more information about scaling the Loggregator system.

Scaling Nozzles

You can scale a nozzle using the subscription ID specified when the nozzle connects to the Firehose. If you use the same subscription ID on each nozzle instance, the Firehose evenly distributes data across all instances of the nozzle.

For example, if you have two nozzle instances with the same subscription ID, the Firehose sends half of the data to one nozzle instance and half to the other. Similarly, if you have three nozzle instances with the same subscription ID, the Firehose sends one-third of the data to each instance.

If you want to scale a nozzle, the number of nozzle instances should match the number of Traffic Controller instances:

Number of nozzle instances = Number of Traffic Controller instances

Stateless nozzles should handle scaling gracefully. If a nozzle buffers or caches the data, the nozzle author must test the results of scaling the number of nozzle instances up or down.

Slow Nozzle Alerts

The Traffic Controller alerts nozzles if they consume events too slowly. If a nozzle falls behind, Loggregator alerts the nozzle in two ways:

- **TruncatingBuffer** alerts: If the nozzle consumes messages more slowly than they are produced, the Loggregator system may drop messages. In this case, Loggregator sends the log message, `TB: Output channel too full. Dropped N messages`, where `N` is the number of dropped messages. Loggregator also emits a **CounterEvent** with the name `doppler_proxy.slow_consumer`. The nozzle receives both messages from the Firehose, alerting the operator to the performance issue.

Forwarding Logs to an External Service

You can configure Pivotal Application Service to forward log data from apps to an external aggregator service. Using Log Management Services explains how to bind apps to the external service and configure it to receive logs from Pivotal Application Service.

Log Message Size Constraints

When a Diego Cell emits app logs to Metron, Diego breaks up log messages greater than approximately 60 KiB into multiple envelopes.

Application Logging in Cloud Foundry

Page last updated:

Loggregator, the Cloud Foundry component responsible for logging, provides a stream of log output from your app and from Cloud Foundry system components that interact with your app during updates and execution.

By default, Loggregator streams logs to your terminal. If you want to persist more than the limited amount of logging information that Loggregator can buffer, you can drain logs to a third-party log management service. See Third-Party Log Management Services.

Cloud Foundry gathers and stores logs in a best-effort manner. If a client is unable to consume log lines quickly enough, the Loggregator buffer may need to overwrite some lines before the client has consumed them. A syslog drain or a CLI tail can usually keep up with the flow of app logs.

Contents of a Log Line

Every log line contains four fields:

1. Timestamp
2. Log type (origin code)
3. Channel: either `OUT` (logs emitted on `stdout`) or `ERR` (logs emitted on `stderr`)
4. Message

Loggregator assigns the timestamp when it receives log data. The log data is opaque to Loggregator, which simply puts it in the message field of the log line. Apps or system components sending log data to Loggregator may include their own timestamps, which then appear in the message field.

Origin codes distinguish the different log types. Origin codes from system components have three letters. The app origin code is `APP` followed by slash and a digit that indicates the app instance.

Many frameworks write to an app log that is separate from STDOUT and STDERR. This is not supported by Loggregator. Your app must write to STDOUT or STDERR for its logs to be included in the Loggregator stream. Check the buildpack your app uses to determine whether it automatically ensures that your app correctly writes logs to STDOUT and STDERR only. Some buildpacks do this, and some do not.

Log Types and Their Messages

Different types of logs have different message formats, as shown in the examples below. The digit appended to the code indicates the instance index: 0 is the first instance, 1 is the second, and so on.

API

Users make API calls to request changes in app state. Cloud Controller, the Cloud Foundry component responsible for the API, logs the actions that Cloud Controller takes in response.

For example:

```
2016-06-14T14:10:05.36-0700 [API/0]     OUT Updated app with guid cdabc600-0b73-48e1-b7d2-26af2c63f933 ({"name"=>"spring-music", "instances"=>1, "memory"=>512, "environment_js
```

STG

The Diego cell or the Droplet Execution Agent emits STG logs when staging or restaging an app. These actions implement the desired state requested by the user. After the droplet has been uploaded, STG messages end and CELL messages begin. For STG, the instance index is almost always 0.

For example:

```
2016-06-14T14:10:27.91-0700 [STG/0]     OUT Staging...
```

RTR

The Router emits RTR logs when it routes HTTP requests to the app. Router messages include the app name followed by a Router timestamp and then selections from the HTTP request.

For example:

```
2016-06-14T10:51:32.51-0700 [RTR/1]     OUT www.example.com - [14/06/2016:17:51:32.459 +0000] "GET /user/ HTTP/1.1" 200 0 103455 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64) Ap
```

Zipkin Trace Logging

If Zipkin trace logging is enabled in Cloud Foundry, then Gorouter access log messages contain Zipkin HTTP headers.

The following is an example access log message containing Zipkin headers:

```
2016-11-23T16:04:01.49-0800 [RTR/0]     OUT www.example.com - [24/11/2016:00:04:01.227 +0000] "GET / HTTP/1.1" 200 0 109 "-" "curl/7.43.0" 10.0.2.150:4070 10.0.48.66:60815 x_forw
```

For more information about Zipkin tracing, see the Zipkin Tracking in HTTP Headers topic.

LGR

Loggregator emits LGR to indicate problems with the logging process. Examples include “can’t reach syslog drain url” and “dropped log messages due to high rate.”

APP

Every app emits logs according to choices by the developer.

For example:

```
2016-06-14T14:10:15.18-0700 [APP/0]     OUT Exit status 0
```

SSH

The Diego cell emits SSH logs when a user accesses an application container through SSH by using the cf ssh command.

For example:

```
2016-06-14T14:16:11.49-0700 [SSH/0]     OUT Successful remote access by 192.0.2.33:7856
```

CELL

The Diego cell emits CELL logs when it starts or stops the app. These actions implement the desired state requested by the user. The Diego cell also emits messages when an app crashes.

For example:

```
2016-06-14T13:44:38.14-0700 [CELL/0]     OUT Successfully created container
```

Writing to the Log from Your App

Your app must write logs to `stderr` or `stdout`. Both are typically buffered, and you should flush the buffer before delivering the message to Loggregator.

Alternatively, you can write log messages to `stderr` or `stdout` synchronously. This approach is mainly used for debugging because it may affect app performance.

Viewing Logs in the Command Line Interface

You view logs in the CLI using the [cf logs] command. You can tail, dump, or filter log output.

Tailing Logs

`cf logs APP_NAME` streams Loggregator output to the terminal.

For example:

```
$ cf logs spring-music
Connected, tailing logs for app spring-music in org example / space development as admin@example.com...

2016-06-14T15:16:12.70-0700 [RTR/4]     OUT www.example.com - [14/06/2016:22:16:12.582 +0000] "GET / HTTP/1.1" 200 0 103455 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_5
2016-06-14T15:16:20.06-0700 [RTR/4]     OUT www.example.com - [14/06/2016:22:16:20.034 +0000] "GET /test/ HTTP/1.1" 200 0 6879 "http://www.example.com/" "Mozilla/5.0 (Macintosh
2016-06-14T15:16:22.44-0700 [RTR/4]     OUT www.example.com - [14/06/2016:22:17:22.415 +0000] "GET /test5/ HTTP/1.1" 200 0 5461 "http://www.example.com/test5" "Mozilla/5.0 (Mac
   ...
```

Use **Ctrl-C** (^C) to exit the real-time stream.

Dumping Logs

`cf logs APP_NAME --recent` displays all the lines in the Loggregator buffer.

Filtering Logs

To view some subset of log output, use `cf logs` in conjunction with filtering commands of your choice. In the example below, `grep -v` excludes all Router logs:

```
$ cf logs spring-music --recent | grep -v RTR
2016-06-14T14:10:05.36-0700 [API/0]     OUT Updated app with guid cdabc604-0b73-47e1-a7d5-24af2c63f723 ({"name"=>"spring-music", "instances"=>1, "memory"=>512, "environment_js
2016-06-14T14:10:14.52-0700 [APP/0]     OUT - Gracefully stopping, waiting for requests to finish
2016-06-14T14:10:14.52-0700 [CELL/0]    OUT Exit status 0
2016-06-14T14:10:14.54-0700 [APP/0]     OUT === puma shutdown: 2016-06-14 21:10:14 +0000 ===
2016-06-14T14:10:14.54-0700 [APP/0]     OUT - Goodbye!
2016-06-14T14:10:14.56-0700 [CELL/0]    OUT Creating container
   ...
```

Security Event Logging for Cloud Controller and UAA

Page last updated:

This topic describes how to enable and interpret security event logging for the Cloud Controller and the User Account and Authentication (UAA) server. Operators can use these logs to retrieve information about a subset of requests to the Cloud Controller and the UAA server for security or compliance purposes.

Cloud Controller Logging

The Cloud Controller logs security events to syslog. You must configure a syslog drain to forward your system logs to a log management service.

See the Configuring System Logging in PAS topic for more information.

Format for Log Entries

Cloud Controller logs security events in the Common Event Format (CEF). CEF specifies the following format for log entries:

```
CEF:Version|Device Vendor|Device Product|Device Version|Signature ID|Name|Severity|Extension
```

Entries in the Cloud Controller log use the following format:

```
CEF:CEF_VERSION|cloud_foundry|cloud_controller_ng|CC_API_VERSION|
SIGNATURE_ID|NAME|SEVERITY|rt=TIMESTAMP suser=USERNAME suid=USER_GUID
cs1Label=userAuthenticationMechanism cs1=AUTH_MECHANISM
cs2Label=vcapRequestId cs2=VCAP_REQUEST_ID request=REQUEST
requestMethod=REQUEST_METHOD cs3Label=result cs3=RESULT
cs4Label=httpStatusCode cs4=HTTP_STATUS_CODE src=SOURCE_ADDRESS
dst=DESTINATION_ADDRESS cs5Label=xForwardedFor cs5=X_FORWARDED_FOR_HEADER
```

Refer to the following list for a description of the properties shown in the Cloud Controller log format:

- `CEF_VERSION` : The version of CEF used in the logs.
- `CC_API_VERSION` : The current Cloud Controller API version.
- `SIGNATURE_ID` : The method and path of the request. For example, `GET /v2/app:GUID` .
- `NAME` : The same as `SIGNATURE_ID` .
- `SEVERITY` : An integer that reflects the importance of the event.
- `TIMESTAMP` : The number of milliseconds since the Unix epoch.
- `USERNAME` : The name of the user who originated the request.
- `USER_GUID` : The GUID of the user who originated the request.
- `AUTH_MECHANISM` : The user authentication mechanism. This can be `oauth-access-token` , `basic-auth` , or `no-auth` .
- `VCAP_REQUEST_ID` : The VCAP request ID of the request.
- `REQUEST` : The request path and parameters. For example, `/v2/info?MY-PARAM=VALUE` .
- `REQUEST_METHOD` . The method of the request. For example, `GET` .
- `RESULT` : The meaning of the HTTP status code of the response. For example, `success` .
- `HTTP_STATUS_CODE` . The HTTP status code of the response. For example, `200` .
- `SOURCE_ADDRESS` : The IP address of the client who originated the request.
- `DESTINATION_ADDRESS` : The IP address of the Cloud Controller VM.
- `X_FORWARDED_FOR_HEADER` : The contents of the X-Forwarded-For header of the request. This is empty if the header is not present

Example Log Entries

The following list provides several example requests with the corresponding Cloud Controller log entries.

- An anonymous `GET` request:

```
CEF:0|cloud_foundry|cloud_controller_ng|2.54.0|GET /v2/info|GET
/v2/info|0|rt=1460690037402 suser= suid= request=/v2/info
requestMethod=GET src=127.0.0.1 dst=192.0.2.1
cs1Label=userAuthenticationMechanism cs1=no-auth cs2Label=vcapRequestId
cs2=c4bac383-7cc9-4d9f-b1c0-1iq8c0baa000 cs3Label=result cs3=success
cs4Label=httpStatusCode cs4=200 cs5Label=xForwardedFor
cs5=198.51.100.1
```

- A `GET` request with basic authentication:

```
CEF:0|cloud_foundry|cloud_controller_ng|2.54.0|GET /v2/syslog_drain_urls|GET
/v2/syslog_drain_urls|0|rt=1460690165743 suser=bulk_api suid=
request=/v2/syslog_drain_urls?batch_size\=1000 requestMethod=GET
src=127.0.0.1 dst=192.0.2.1 cs1Label=userAuthenticationMechanism
cs1=basic-auth cs2Label=vcapRequestId cs2=79187189-e810-33dd-6911-b5d015bbc999
::eat1234d-4004-4622-ad11-9iaai88e3ae9 cs3Label=result cs3=success
cs4Label=httpStatusCode cs4=200 cs5Label=xForwardedFor cs5=198.51.100.1
```

- A `GET` request with OAuth access token authentication:

```
CEF:0|cloud_foundry|cloud_controller_ng|2.54.0|GET /v2/routes|GET
/v2/routes|0|rt=1460689904925 suser=admin suid=c7ca208f-8a9e-4aab-
92f5-28795f86d62a request=/v2/routes?inline-relations-depth\=1&q\=
host%3Adora%3Bdomain_guid%3B777-1o9f-5f5n-i888-o2025cb2dfc3
requestMethod=GET src=127.0.0.1 dst=192.0.2.1
cs1Label=userAuthenticationMechanism cs1=oauth-access-token
cs2Label=vcapRequestId cs2=79187189-990i-8930-52b2-9090b2c5poz0
::5a265621-b223-4520-afae-ab7d0ee7c75b cs3Label=result cs3=success
cs4Label=httpStatusCode cs4=200 cs5Label=xForwardedFor cs5=198.51.100.1
```

- A `GET` request that results in a 404 error:

```
CEF:0|cloud_foundry|cloud_controller_ng|2.54.0|GET /v2/apps/7f310103-
39aa-4a8c-b92a-9ff8a6a2fa6b|GET /v2/apps/7f310103-39aa-4a8c-b92a-
9ff8a6a2fa6b|0|rt=1460691002394 suser=bob suid=a00i2026-55io-3983-
555o-40e611410aec request=/v2/apps/7f310103-39aa-4a8c-b92a-9ff8a6a2fa6b
requestMethod=GET src=127.0.0.1 dst=192.0.2.1
cs1Label=userAuthenticationMechanism cs1=oauth-access-token cs2Label=vcapRequestId
cs2=49f21579-9eb5-4bdf-6e49-e77d2de647a2::9f8841e6-e04a-498b-b3ff-d59cfe7cb7ea
cs3Label=result cs3=clientError cs4Label=httpStatusCode cs4=404
cs5Label=xForwardedFor cs5=198.51.100.1
```

- A `POST` request that results in a 403 error:

```
CEF:0|cloud_foundry|cloud_controller_ng|2.54.0|POST /v2/apps|POST
/v2/apps|0|rt=1460691405564 suser=bob suid=4f9a33f9-fb13-4774-a708-
f60c939625cd request=/v2/apps?async\=true requestMethod=POST
src=127.0.0.1 dst=192.0.2.1 cs1Label=userAuthenticationMechanism
cs1=oauth-access-token cs2Label=vcapRequestId cs2=booc03111-9999-4003-88ab-
20i9r33333ou::5a4993fc-722f-48bc-aff4-99b2005i9bb5 cs3Label=result
cs3=clientError cs4Label=httpStatusCode cs4=403 cs5Label=xForwardedFor
cs5=198.51.100.1
```

UAA Logging

UAA logs security events to a file located at `/var/vcap/sys/log/uaa/uaa.log` on the UAA virtual machine (VM). Because these logs are automatically rotated, you must configure a syslog drain to forward your system logs to a log management service.

See the Configuring System Logging in PAS topic for more information.

Log Events

UAA logs identify the following categories of events:

- Authorization and Password Events
- SCIM Administration Events

- Token Events
- Client Administration Events
- UAA Administration Events

To learn more about the names of the events included in these categories and the information they record in the UAA logs, see User Account and Authentication Service Audit Requirements.

Example Log Entries

The following sections provide several example requests with the corresponding UAA log entries.

Successful User Authentication

```
Audit: TokenIssuedEvent ('["openid","scim.read","uaa.user",
"cloud_controller.read","password.write","cloud_controller.write",
"scim.write"]'): principal=a42026d6-5533-1884-eef2-838abcd0i3e3,
origin=[client=cf, user=bob], identityZoneId=[uaa]
```

- This entry records a `TokenIssuedEvent`.
- UAA issued a token associated with the scopes `"openid","scim.read","uaa.user", "cloud_controller.read","password.write","cloud_controller.write","scim.write"` to the user `bob`.

Failed User Authentication

```
Audit: UserAuthenticationFailure ('bob@example.com'):
principal=61965469-c821-46b7-825f-630e12a51d6c,
origin=[remoteAddress=198.51.100.1, clientId=cf],
identityZoneId=[uaa]
```

- This entry records a `UserAuthenticationFailure`.
- The user `bob@example.com` originating at `198.51.100.1` failed to authenticate.

Successful User Creation

```
Audit: UserCreatedEvent ('["user_id=61965469-c821-
46b7-825f-630e12a51d6c","username=bob@example.com"]'):
principal=91220262-d901-44c0-825f-633i33b55d6c,
origin=[client=cf, user=admin, details=(198.51.100.1,
tokenType=bearertokenValue=<TOKEN>,
sub=20i03423-dd8e-33e1-938d-e9999e30f500,
iss=https://uaa.example.com/oauth/token)], identityZoneId=[uaa]
```

- This entry records a `UserCreatedEvent`.
- The `admin` user originating at `198.51.100.1` created a user named `bob@example.com`.

Successful User Deletion

```
Audit: UserDeletedEvent ('["user_id=61965469-c821-
46b7-825f-630e12a51d6c","username=bob@example.com"]'):
principal=61965469-c821-46b7-825f-630e12a51d6c,
origin=[client=admin, details=(remoteAddress=198.51.100.1,
tokenType=bearertokenValue=<TOKEN>,
sub=admin, iss=https://uaa.example.com/oauth/token)], identityZoneId=[uaa]
```

- This entry records a `UserDeletedEvent`.
- The `admin` user originating at `198.51.100.1` deleted a user named `bob@example.com`.

Deploying a Nozzle to the Loggregator Firehose

Page last updated:

This topic describes deploying a nozzle application to the Cloud Foundry (CF) Loggregator Firehose. The Cloud Foundry Loggregator team created an example nozzle application for use with this tutorial.

The procedure described below deploys this example nozzle to the Firehose of a Cloud Foundry installation deployed locally with BOSH Lite. For more information about BOSH Lite, see the BOSH Lite GitHub repository.

Prerequisites

- BOSH CLI installed locally.
- Spiff installed locally and added to the load path of your shell. See Spiff on GitHub.
- BOSH Lite deployed locally using VirtualBox. See BOSH Lite on GitHub.
- A working Cloud Foundry deployment, including Loggregator, deployed with your local BOSH Lite. This serves as our source of data. See Deploying Cloud Foundry using BOSH Lite, or use the `provision_cf` script included in the BOSH Lite release.

> **Note**: Deploying Cloud Foundry can take up to several hours, depending on your internet bandwith, even when using the automated `provision_cf` script.

Step 1: Download Cloud Foundry BOSH Manifest

1. Run `bosh -e MY-ENV deployments` to identify the names of all deployments in the environment you specify. Replace `MY-ENV` with the alias you set for your BOSH Director. For example:

```
$ bosh -e dev deployments
Using environment '192.168.15.4' as client 'admin'

Name                     Release(s)             Stemcell(s)                              Team(s)  Cloud Config
cf-example               binary-buildpack/1.0.9  bosh-warden-boshlite-ubuntu-trusty-go_agent/3363.9  -   latest
                         capi/1.21.0
                         cf-mysql/34
                         cf-smoke-tests/11
                         cflinuxfs2-rootfs/1.52.0
                         consul/155
                         diego/1.8.1
                         etcd/94
                         garden-runc/1.2.0
                         loggregator/78
                         nats/15
                         routing/0.145.0
                         statsd-injector/1.0.20
                         uaa/25
service-instance_0d4140a0-42b7-...  mysql/0.6.0    bosh-warden-boshlite-ubuntu-trusty-go_agent/3363.9  -   latest

2 deployments

Succeeded
```

2. Run `bosh -e MY-ENV -d MY-DEPLOYMENT manifest > MY-MANIFEST.yml` to download and save the current BOSH deployment manifest. Replace `MY-ENV` with your BOSH Director alias, `MY-DEPLOYMENT` with the deployment name from the output of the previous step, and `MY-MANIFEST.yml` with a name you choose for the saved manifest file. You need this manifest to locate information about your databases.

```
$ bosh -e dev -d cf-example manifest cf.yml
```

Step 2: Add UAA client

You must authorize the example nozzle as a UAA client for your CF deployment. To do this, add an entry for the example nozzle as `client` for `uaa` under the `properties` key in your CF deployment manifest. You must enter the example nozzle object in the correct location in the manifest, and with the correct indentation, as described below.

Deployment manifests are YAML files. Visit YAML to learn about YAML syntax.

1. Open the deployment manifest in a text editor.
2. Locate the left-aligned `properties` key.
3. Under the `properties` key, locate `uaa` at the next level of indentation.
4. Under the `uaa` key, locate the `clients` key at the next level of indentation.
5. Enter properties for the `example-nozzle` at the next level of indentation, exactly as shown below. The `...` in the text below indicate other properties that may populate the manifest at each level in the hierarchy.

```
properties:
  ...
  uaa:
  ...
    clients:
    ...
      example-nozzle:
        access-token-validity: 1209600
        authorized-grant-types: client_credentials
        override: true
        secret: example-nozzle
        authorities: oauth.login,doppler.firehose
```

6. Save the deployment manifest file.

Step 3: Redeploy Cloud Foundry

1. Deploy Cloud Foundry with BOSH.

```
$ bosh deploy
Acting as user 'admin' on deployment 'cf-warden' on 'Bosh Lite Director'
Getting deployment properties from director...

Detecting deployment changes
---------------------------
Releases
No changes

Compilation
No changes

Update
No changes

Resource pools
No changes

Disk pools
No changes

Networks
No changes

Jobs
No changes

Properties
uaa
clients
example-nozzle
  + access-token-validity: 1209600
  + authorized-grant-types: authorization_code,client_credentials,refresh_token
  + override: true
  + secret: example-nozzle
  + scope: openid,oauth.approvals,doppler.firehose
  + authorities: oauth.login,doppler.firehose

Meta
No changes

Please review all changes carefully

Deploying
---------
Are you sure you want to deploy? (type 'yes' to continue):yes
```

Step 4: Clone Example Release

The Cloud Foundry Loggregator team created an example nozzle application for use with this tutorial.

1. Run `git clone` to clone the main release repository from GitHub.

```
$ git clone git@github.com:cloudfoundry-incubator/example-nozzle-release.git
Cloning into 'example-nozzle-release'...
```

2. Run `git submodule update --init --recursive` to update all of the included submodules.

```
$ git submodule update --init --recursive
Submodule 'src/github.com/cloudfoundry-incubator/example-nozzle' (git@github.com:cloudfoundry-incubator/example-nozzle.git) registered for path 'src/github.com/cloudfoundry-incub
Submodule 'src/github.com/cloudfoundry-incubator/uaago' (git@github.com:cloudfoundry-incubator/uaago.git) registered for path 'src/github.com/cloudfoundry-incubator/uaago'
...
Cloning into 'src/github.com/cloudfoundry-incubator/example-nozzle'...
...
```

3. Navigate to the `example-release` directory.

```
$ cd example-nozzle-release
```

Step 5: Prepare Nozzle Manifest

Complete the following steps to prepare the nozzle deployment manifest:

1. In the `example-nozzle-release` directory, navigate to the `templates` directory.

```
$ cd templates
```

   Within this directory, examine the two YAML files. `bosh-lite-stub.yml` contains the values used to populate the missing information in `template.yml`. By combining these two file, we create a deployment manifest for our nozzle.

2. Create a `tmp` directory for the compiled manifest.

3. Use Spiff to compile a deployment manifest from the template and stub, and save this manifest.

```
$ spiff merge templates/template.yml templates/bosh-lite-stub.yml > tmp/manifest_bosh_lite.yml
```

4. Run `bosh env --uuid` to obtain your BOSH director UUID.

```
$ bosh env --uuid
```

5. In the compiled nozzle deployment manifest, locate the `director_uuid` property. Replace `PLACEHOLDER-DIRECTOR-UUID` with your BOSH director UUID.

```
compilation:
cloud_properties:
  name: default
network: example-nozzle-net
reuse_compilation_vms: true
workers: 1
director_uuid: PLACEHOLDER-DIRECTOR-UUID  # replace this
```

Note: If you do not want to see the complete deployment procedure, run the following command to automatically prepare the manifest: `scripts/make_manifest_spiff_bosh_lite`

Step 6: Create Nozzle BOSH Release

Use the `bosh create-release --name RELEASE-NAME` command to create a BOSH release. Replace RELEASE-NAME with `example-nozzle` to match the UAA client that you created in the CF deployment manifest.

```
$ bosh  --name example-nozzle
Syncing blobs...
...
```

Step 9: Upload Nozzle BOSH Release

Run `bosh upload-release` to upload the release that you created in Step 7: Create Nozzle BOSH Release.


```
$ bosh upload-release
Acting as user 'admin' on 'Bosh Lite Director'

Copying packages
----------------
example-nozzle
golang1.7

Copying jobs
------------
example-nozzle

Generated   /var/folders/4n/qs1rjbmd1c5gfb78m3_06j6r0000gn/T/d20151009-71219-17a5m49/d20151009-71219-rts928/release.tgz
Release size: 59.2M

Verifying release...
...
Release info
------------
Name: nozzle-test
Version: 0+dev.2

Packages
- example-nozzle (b0944f95eb5a332e9be2adfb4db1bc88f9755894)
- golang1.7 (b68dc9557ef296cb21e577c31ba97e2584a5154b)

Jobs
- example-nozzle (112e01c6ee91e8b268a42239e58e8e18e0360f58)

License
- none

Uploading release
```

Step 10: Deploy Nozzle

Run `bosh deploy` to deploy the nozzle.

```
$ bosh deploy
Acting as user 'admin' on deployment 'example-nozzle-lite' on 'Bosh Lite Director'
Getting deployment properties from director...
Unable to get properties list from director, trying without it...
Cannot get current deployment information from director, possibly a new deployment
Please review all changes carefully

Deploying
---------
Are you sure you want to deploy? (type 'yes' to continue):yes
```

Step 11: View Nozzle Output

The example nozzle outputs all of the data originating coming from the Firehose to its log files. To view this data, SSH into the example-nozzle VM and examine the logs.

1. Run `bosh ssh` to access the nozzle VM at the IP configured in the nozzle’s manifest template stub `./templates/bosh-lite-stub.yml`.

   ```
   $ bosh ssh example-nozzle
   Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.19.0-25-generic x86_64)
   Documentation: https://help.ubuntu.com/
   Last login: Wed Sep 23 21:29:50 2015 from 192.0.2.1
   ```

2. Use the `cat` command to output the `stdout` log file.

```
$ cat /var/vcap/sys/log/example-nozzle/example-nozzle.stdout.log
===== Streaming Firehose (will only succeed if you have admin credentials)
origin:"DopplerServer" eventType:ValueMetric timestamp:1443046217700750747 deployment:"cf-warden" job:"doppler_z1" index:"0" ip:"203.0.113.142" valueMetric:
origin:"MetronAgent" eventType:CounterEvent timestamp:1443046218910193187 deployment:"cf-warden" job:"loggregator_trafficcontroller_z1" index:"0" ip:"203.0.113.146" counterE
origin:"MetronAgent" eventType:CounterEvent timestamp:1443046218910360012 deployment:"cf-warden" job:"loggregator_trafficcontroller_z1" index:"0" ip:"203.0.113.146" counterE
origin:"MetronAgent" eventType:CounterEvent timestamp:1443046218910252169 deployment:"cf-warden" job:"loggregator_trafficcontroller_z1" index:"0" ip:"203.0.113.146" counterE
origin:"MetronAgent" eventType:CounterEvent timestamp:1443046218910294255 deployment:"cf-warden" job:"loggregator_trafficcontroller_z1" index:"0" ip:"203.0.113.146" counterE
origin:"MetronAgent" eventType:CounterEvent timestamp:1443046218910318582 deployment:"cf-warden" job:"loggregator_trafficcontroller_z1" index:"0" ip:"203.0.113.146" counterE
origin:"MetronAgent" eventType:CounterEvent timestamp:1443046218910339088 deployment:"cf-warden" job:"loggregator_trafficcontroller_z1" index:"0" ip:"203.0.113.146" counterE
origin:"MetronAgent" eventType:CounterEvent timestamp:1443046218910379199 deployment:"cf-warden" job:"loggregator_trafficcontroller_z1" index:"0" ip:"203.0.113.146" counterE
origin:"MetronAgent" eventType:CounterEvent timestamp:1443046218910394886 deployment:"cf-warden" job:"loggregator_trafficcontroller_z1" index:"0" ip:"203.0.113.146" counterE
origin:"router__0" eventType:HttpStartStop timestamp:1443046219105062148 deployment:"cf-warden" job:"router_z1" index:"0" ip:"203.0.113.22" httpStartStop: peerType:Client metho
origin:"api_z1_0" eventType:HttpStartStop timestamp:1443046219109842455 deployment:"cf-warden" job:"api_z1" index:"0" ip:"203.0.113.134" httpStartStop: peerType:Server method
origin:"router__0" eventType:HttpStartStop timestamp:1443046219110064368 deployment:"cf-warden" job:"router_z1" index:"0" ip:"203.0.113.22" httpStartStop: peerType:Client metho
origin:"syslog_drain_binder" eventType:ValueMetric timestamp:1443046219177165446 deployment:"cf-warden" job:"doppler_z1" index:"0" ip:"203.0.113.142" valueMetric:
origin:"syslog_drain_binder" eventType:ValueMetric timestamp:1443046219177288325 deployment:"cf-warden" job:"doppler_z1" index:"0" ip:"203.0.113.142" valueMetric:
origin:"syslog_drain_binder" eventType:ValueMetric timestamp:1443046219177346726 deployment:"cf-warden" job:"doppler_z1" index:"0" ip:"203.0.113.142" valueMetric:
origin:"syslog_drain_binder" eventType:ValueMetric timestamp:1443046219177274975 deployment:"cf-warden" job:"doppler_z1" index:"0" ip:"203.0.113.142" valueMetric:
origin:"syslog_drain_binder" eventType:ValueMetric timestamp:1443046219177310389 deployment:"cf-warden" job:"doppler_z1" index:"0" ip:"203.0.113.142" valueMetric:
origin:"syslog_drain_binder" eventType:ValueMetric timestamp:1443046219177330214 deployment:"cf-warden" job:"doppler_z1" index:"0" ip:"203.0.113.142" valueMetric:
origin:"syslog_drain_binder" eventType:ValueMetric timestamp:1443046219177353454 deployment:"cf-warden" job:"doppler_z1" index:"0" ip:"203.0.113.142" valueMetric:
origin:"syslog_drain_binder" eventType:ValueMetric timestamp:1443046219177360052 deployment:"cf-warden" job:"doppler_z1" index:"0" ip:"203.0.113.142" valueMetric:
origin:"syslog_drain_binder" eventType:ValueMetric timestamp:1443046219177481456 deployment:"cf-warden" job:"doppler_z1" index:"0" ip:"203.0.113.142" valueMetric:
origin:"DopplerServer" eventType:CounterEvent timestamp:1443046219880585603 deployment:"cf-warden" job:"doppler_z1" index:"0" ip:"203.0.113.142" counterEvent:
origin:"DopplerServer" eventType:CounterEvent timestamp:1443046219880895040 deployment:"cf-warden" job:"doppler_z1" index:"0" ip:"203.0.113.146" counterEvent:
origin:"DopplerServer" eventType:CounterEvent timestamp:1443046219881017888 deployment:"cf-warden" job:"doppler_z1" index:"0" ip:"203.0.113.142" counterEvent:
origin:"DopplerServer" eventType:CounterEvent timestamp:1443046219881011670 deployment:"cf-warden" job:"doppler_z1" index:"0" ip:"203.0.113.142" counterEvent:
origin:"DopplerServer" eventType:CounterEvent timestamp:1443046219880929574 deployment:"cf-warden" job:"doppler_z1" index:"0" ip:"203.0.113.142" counterEvent:
origin:"DopplerServer" eventType:CounterEvent timestamp:1443046219881004417 deployment:"cf-warden" job:"doppler_z1" index:"0" ip:"203.0.113.142" counterEvent:
origin:"DopplerServer" eventType:CounterEvent timestamp:1443046219880929568 deployment:"cf-warden" job:"doppler_z1" index:"0" ip:"203.0.113.142" counterEvent:
origin:"MetronAgent" eventType:CounterEvent timestamp:1443046220058280679 deployment:"cf-warden" job:"api_z1" index:"0" ip:"203.0.113.134" counterEvent:
```

Logs and Metrics Sources

Page last updated:

Cloud Foundry logs and metrics come from several sources.

- Loggregator is the next generation logging and metrics system for Cloud Foundry (CF). Loggregator aggregates metrics from applications and CF system components and streams these out to the Cloud Foundry Command Line Interface (cf CLI) or to third-party log management services. See Loggregator for Operators for more information.
- CF components can also forward syslog logs in syslog format directly to a syslog drain, bypassing Loggregator.
- The BOSH Health Monitor continually listens for one **heartbeat** per minute from each deployed virtual machine (VM). These heartbeats contain status updates and lifecycle events. Health Monitor can be extended by plugins to forward heartbeat data to other CF components or third-party services.

Cloud Foundry supports all of these metrics pipelines. Data from each of these sources can be streamed to a variety of services including the following:

- JMX Bridge
- Datadog
- AWS CloudWatch

See Using Log Management Services for more information about draining logs from Pivotal Application Service.

Installing the Loggregator Firehose Plugin for cf CLI

Page last updated:

The Loggregator Firehose plugin for the Cloud Foundry Command Line Interface (cf CLI) allows Cloud Foundry (CF) administrators access to the output of the Loggregator Firehose, which includes logs and metrics from all CF components.

See Using cf CLI Plugins for more information about using plugins with the cf CLI.

Prerequisites

- Administrator access to the Cloud Foundry deployment that you want to monitor
- Cloud Foundry Command Line Interface (cf CLI) 6.12.2 or later

Refer to the Installing the cf CLI topic for information about downloading, installing, and uninstalling the cf CLI.

Install the Plugin

1. Run `cf add-plugin-repo REPOSITORY-NAME URL` to add the CF Community plugin repository to your cf CLI plugins.

   ```
   $ cf add-plugin-repo CF-Community https://plugins.cloudfoundry.org
   ```

2. Run `cf install-plugin -r PLUGIN-REPOSITORY PLUGIN-NAME` to install the Firehose plugin from the CF Community plugin repository.

   ```
   $ cf install-plugin -r CF-Community "Firehose Plugin"
   ```

View the Firehose

Run `cf nozzle --debug` to view the streaming output of the Firehose, which includes logging events and metrics from CF system components. For more information about logging and metrics in CF, see Overview of the Loggregator System.

```
$ cf nozzle --debug
```

Note: You must be logged in as a Cloud Foundry administrator to access the Firehose.

Uninstall the Plugin

Run `cf plugins` to see a list of installed plugins.

```
$ cf plugins
Listing Installed Plugins...
OK
Plugin Name     Version   Command Name   Command Help
FirehosePlugin  0.6.0     nozzle         Command to print out messages from the firehose
```

Run `cf uninstall-plugin PLUGIN-NAME` to uninstall the plugin.

```
$ cf uninstall-plugin FirehosePlugin
```

Troubleshooting and Diagnostics

Navigate these topics to troubleshoot and diagnose a variety of problems you may encounter in PCF.

Table of Contents

Diagnosing Problems in PCF

Page last updated:

This guide provides help with diagnosing issues encountered during a Pivotal Cloud Foundry (PCF) installation.

Besides whether products install successfully or not, an important area to consider when diagnosing issues is communication between VMs deployed by Pivotal Cloud Foundry. Depending on what products you install, communication takes the form of messaging, routing, or both. If they go wrong, an installation can fail. For example, in an Pivotal Application Service (PAS) installation the PCF VM tries to push a test application to the cloud during post-installation testing. The installation fails if the resulting traffic cannot be routed to the HA Proxy load balancer.

Viewing the Debug Endpoint

The debug endpoint is a web page that provides information useful for diagnostics. If you have superuser privileges and can view the Ops Manager Installation Dashboard, you can access the debug endpoint.

- In a browser, open the URL:
  `https://OPS-MANAGER-FQDN/debug`

The debug endpoint offers three links:

- *Files* allows you to view the YAML files that Ops Manager uses to configure products that you install. The most important YAML file, `installation.yml`, provides networking settings and describes `microbosh`. In this case, `microbosh` is the VM whose BOSH Director component is used by Ops Manager to perform installations and updates of PAS and other products.
- *Components* describes the components in detail.
- *Rails log* shows errors thrown by the VM where the Ops Manager web application (a Rails application) is running, as recorded in the `production.log` file. See the next section to learn how to explore other logs.

Logging Tips

Identifying Where to Start

This section contains general tips for locating where a particular problem is called out in the log files. Refer to the later sections for tips regarding specific logs (such as those for PAS Components).

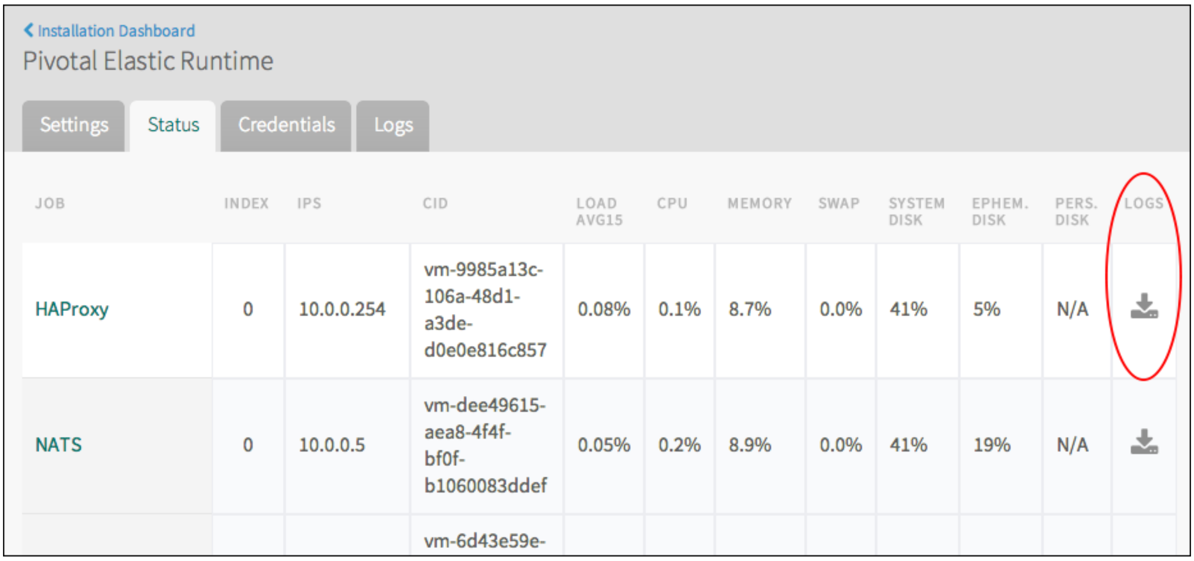
- Start with the largest and most recently updated files in the job log
- Identify logs that contain 'err' in the name
- Scan the file contents for a "failed" or "error" string

Viewing Logs for PAS Components

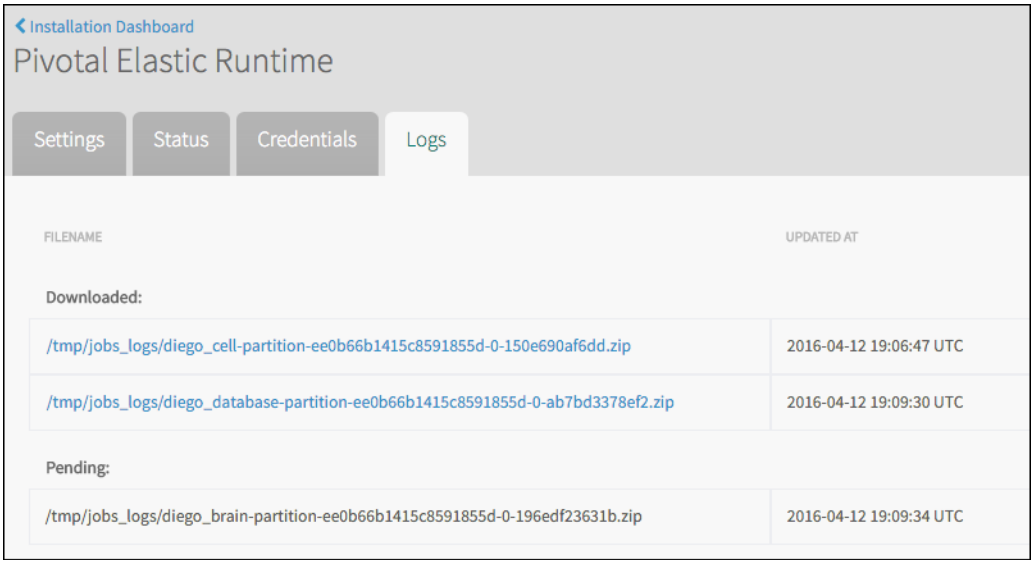
To troubleshoot specific PAS components by viewing their log files, browse to the Ops Manager interface and follow the procedure below.

1. In Ops Manager, browse to the PAS **Status** tab. In the **Job** column, locate the component of interest.

2. In the **Logs** column for the component, click the download icon.



3. Browse to the PAS **Logs** tab.



4. Once the zip file corresponding to the component of interest moves to the **Downloaded** list, click the linked file path to download the zip file.

5. Once the download completes, unzip the file.

The contents of the log directory vary depending on which component you view. For example, the Diego cell log directory contains subdirectories for the `metron_agent`, `rep`, `monit`, and `garden` processes. To view the standard error stream for `garden`, download the Diego cell logs and open `diego.0.job > garden > garden.stderr.log`.

Viewing Web Application and BOSH Failure Logs in a Terminal Window

You can obtain diagnostic information from the Operations Manager by logging in to the VM where it is running. To log in to the Operations Manager VM, you need the following information:

- The IP address of the PCF VM shown in the `Settings` tab of the Ops Manager Director tile.
- Your **import credentials**. Import credentials are the username and password used to import the PCF `.ova` or `.ovf` file into your virtualization system.

Complete the following steps to log in to the Operations Manager VM:

1. Open a terminal window.

2. Run `ssh IMPORT-USERNAME@PCF-VM-IP-ADDRESS` to connect to the PCF installation VM.

3. Enter your import password when prompted.

4. Change directories to the home directory of the web application:
   `cd /home/tempest-web/tempest/web/`

5. You are now in a position to explore whether things are as they should be within the web application.

   You can also verify that the `microbosh` component is successfully installed. A successful MicroBOSH installation is required to install PAS and any products like databases and messaging services.

6. Change directories to the BOSH installation log home:
   `cd /var/tempest/workspaces/default/deployments/micro`

7. You may want to begin by running a tail command on the `current` log:
   `cd /var/tempest/workspaces/default/deployments/micro`
   If you are unable to resolve an issue by viewing configurations, exploring logs, or reviewing common problems, you can troubleshoot further by running BOSH diagnostic commands with the BOSH Command Line Interface (CLI).

Note: Do not manually modify the deployment manifest. Operations Manager will overwrite manual changes to this manifest. In addition, manually changing the manifest may cause future deployments to fail.

Viewing the VMs in Your Deployment

To view the VMs in your PCF deployment, perform the following steps specific to your IaaS.

Amazon Web Services (AWS)

1. Log in to the AWS Console.

2. Navigate to the EC2 Dashboard.

3. Click **Running Instances**.

4. Click the gear icon in the upper right.

5. Select the following: **job**, **deployment**, **director**, **index**.

6. Click **Close**.

OpenStack

1. Install the novaclient.

2. Point novaclient to your OpenStack installation and tenant by exporting the following environment variables:

```
$ export OS_AUTH_URL= YOUR_KEYSTONE_AUTH_ENDPOINT
$ export OS_TENANT_NAME = TENANT_NAME
$ export OS_USERNAME = USERNAME
$ export OS_PASSWORD = PASSWORD
```

3. List your VMs by running the following command:

```
$ nova list --fields metadata
```

vSphere

1. Log into vCenter.

2. Select **Hosts and Clusters**.
3. Select the top level object that contains your PCF deployment. For example, select **Cluster**, **Datastore** or **Resource Pool**.
4. In the top tab, click **Related Objects**.
5. Select **Virtual Machines**.
6. Right click on the **Table** heading and select **Show/Hide Columns**.
7. Select the following boxes: **job**, **deployment**, **director**, **index**.

Viewing Apps Manager Logs in a Terminal Window

The Apps Manager provides a graphical user interface to help manage organizations, users, applications, and spaces.

When troubleshooting Apps Manager performance, you might want to view the Apps Manager application logs. To view the Apps Manager application logs, follow these steps:

1. Run `cf login -a api.MY-SYSTEM-DOMAIN -u admin` from a command line to log in to PCF using the UAA Administrator credentials. In Pivotal Ops Manager, refer to PAS **Credentials** for these credentials.

```
$ cf login -a api.example.com -u admin
API endpoint: api.example.com

Password>******
Authenticating...
OK
```

2. Run `cf target -o system -s apps-manager` to target the `system` org and the `apps-manager` space.

```
$ cf target -o system -s apps-manager
```

3. Run `cf logs apps-manager` to tail the Apps Manager logs.

```
$ cf logs apps-manager
Connected, tailing logs for app apps-manager in org system / space apps-manager as
admin...
```

Changing Logging Levels for the Apps Manager

The Apps Manager recognizes the `LOG_LEVEL` environment variable. The `LOG_LEVEL` environment variable allows you to filter the messages reported in the Apps Manager log files by severity level. The Apps Manager defines severity levels using the Ruby standard library Logger class.

By default, the Apps Manager `LOG_LEVEL` is set to `info`. The logs show more verbose messaging when you set the `LOG_LEVEL` to `debug`.

To change the Apps Manager `LOG_LEVEL`, run `cf set-env apps-manager LOG_LEVEL` with the desired severity level.

```
$ cf set-env apps-manager LOG_LEVEL debug
```

You can set `LOG_LEVEL` to one of the six severity levels defined by the Ruby Logger class:

- **Level 5:** `unknown` – An unknown message that should always be logged
- **Level 4:** `fatal` – An unhandleable error that results in a program crash
- **Level 3:** `error` – A handleable error condition
- **Level 2:** `warn` – A warning
- **Level 1:** `info` – General information about system operation
- **Level 0:** `debug` – Low-level information for developers

Once set, the Apps Manager log files only include messages at the set severity level and above. For example, if you set `LOG_LEVEL` to `fatal`, the log includes `fatal` and `unknown` level messages only.

Troubleshooting Problems in PCF

Page last updated:

This guide provides help with resolving issues encountered during a Pivotal Cloud Foundry (PCF) installation.

Retrying the Deployment

Although an install or update can fail for many reasons, the system is self-healing, and can often automatically correct or work around hardware or network faults.

Click **Install** or **Apply Changes** again, and the system may resolve a problem on its own.

Some failures only produce generic errors like `Exited with 1`. In cases like this, where a failure is not accompanied by useful information, click **Install** or **Apply Changes** to retry.

When the system does provide informative evidence, review the Common Problems section at the end of this guide to see if your problem is covered there.

Common Issues

Compare evidence that you have gathered to the descriptions below. If your issue is covered, try the recommended remediation procedures.

BOSH Does Not Reinstall

You might want to reinstall BOSH for troubleshooting purposes. However, if PCF does not detect any changes, BOSH does not reinstall. To force a reinstall of BOSH, select **Ops Manager Director > Resource Sizes** and change a resource value. For example, you could increase the amount of RAM by 4 MB.

Creating Bound Missing VMs Times Out

This task happens immediately following package compilation, but before job assignment to agents. For example:

```
cloud_controller/0: Timed out pinging to f690db09-876c-475e-865f-2cece06aba79 after 600 seconds (00:10:24)
```

This is most likely a NATS issue with the VM in question. To identify a NATS issue, inspect the agent log for the VM. Since the BOSH director is unable to reach the BOSH agent, you must access the VM using another method. You will likely also be unable to access the VM using TCP. In this case, access the VM using your virtualization console.

To diagnose:

1. Access the VM using your virtualization console and log in.
2. Navigate to the **Credentials** tab of the Pivotal Application Service (PAS) tile and locate the VM in question to find the **VM credentials**.
3. Become root.
4. Run `cd /var/vcap/bosh/log`.
5. Open the file `current`.
6. First, determine whether the BOSH agent and director have successfully completed a handshake, represented in the logs as a “ping-pong”:

   ```
   2013-10-03_14:35:48.58456 #[608] INFO: Message: {"method"=>"ping", "arguments"=>[],
   "reply_to"=>"director.f4b7df14-cb8f.19719508-e0dd-4f53-b755-58b6336058ab"}

   2013-10-03_14:35:48.60182 #[608] INFO: reply_to:   director.f4b7df14-cb8f.19719508-e0dd-4f53-b755-58b6336058ab:
   payload: {:value=>"pong"}
   ```

   This handshake must complete for the agent to receive instructions from the director.

7. If you do not see the handshake, look for another line near the beginning of the file, prefixed `INFO: loaded new infrastructure settings`. For example:

```
2013-10-03_14:35:21.83222 #[608] INFO: loaded new infrastructure settings:
{"vm"=>{"name"=>"vm-4d80ede4-b0a5-4992-aea6a0386e18e", "id"=>"vm-360"},
"agent_id"=>"56aea4ef-6aa9-4c39-8019-7024ccfdde4",
"networks"=>{"default"=>{"ip"=>"192.0.2.19",
"netmask"=>"255.255.255.0", "cloud_properties"=>{"name"=>"VMNetwork"},
"default"=>["dns", "gateway"],
"dns"=>["192.0.2.2", "192.0.2.17"], "gateway"=>"192.0.2.2",
"dns_record_name"=>"0.nats.default.cf-d729343071061.microbosh",
"mac"=>"00:50:56:9b:71:67"}}, "disks"=>{"system"=>0, "ephemeral"=>1,
"persistent"=>{}}, "ntp"=>[], "blobstore"=>{"provider"=>"dav",
"options"=>{"endpoint"=>"http://192.0.2.17:25250",
"user"=>"agent", "password"=>"agent"}},
"mbus"=>"nats://nats:nats@192.0.2.17:4222",
"env"=>{"bosh"=>{"password"=>"$6$40ftQ9K4rvvC/8ADZHW0"}}}
```

This is a JSON blob of key/value pairs representing the expected infrastructure for the BOSH agent. For this issue, the following section is the most important:

`"mbus"=>"nats://nats:nats@192.0.2.17:4222"`

This key/value pair represents where the agent expects the NATS server to be. One diagnostic tactic is to try pinging this NATS IP address from the VM to determine whether you are experiencing routing issues.

Install Exits With a Creates/Updates/Deletes App Failure or With a 403 Error

Scenario 1: Your PCF install exits with the following 403 error when you attempt to log in to the Apps Manager:

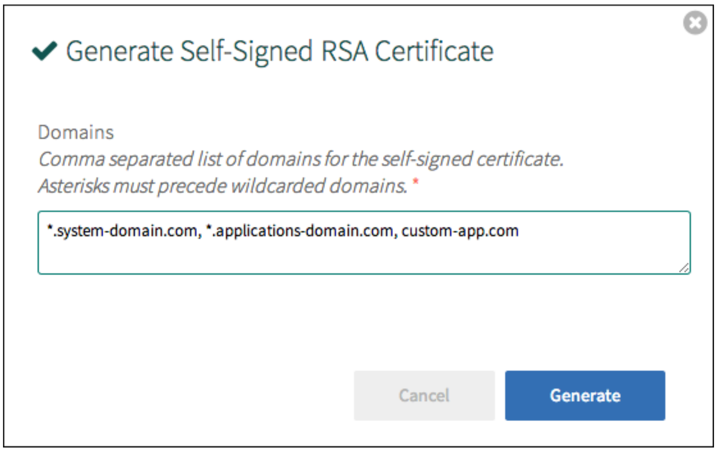
```
{"type": "step_finished", "id": "apps-manager.deploy"}

/home/tempest-web/tempest/web/vendor/bundle/ruby/1.9.1/gems/mechanize-2.7.2/lib/mechanize/http/agent.rb:306:in
`fetch': 403 => Net::HTTPForbidden for https://login.api.example.net/oauth/authorizeresponse_type=code&client_id=portal&redirect_uri=https%3..
-- unhandled response (Mechanize::ResponseCodeError)
```

Scenario 2: Your PCF install exits with a `creates/updates/deletes an app (FAILED - 1)` error message with the following stack trace:

```
1) App CRUD creates/updates/deletes an app
   Failure/Error: Unable to find matching line from backtrace
   CFoundry::TargetRefused:
     Connection refused - connect(2)
```

In either of the above scenarios, ensure that you have correctly entered your domains in wildcard format:

1. Browse to the Operations Manager fully qualified domain name (FQDN).
2. Click the PAS tile.
3. Select **HAProxy** and click **Generate Self-Signed RSA Certificate**.
4. Enter your system and app domains in wildcard format, as well as optionally any custom domains, and click **Save**. Refer to PAS **Cloud Controller** for explanations of these domain values.



Install Fails When Gateway Instances Exceed Zero

If you configure the number of Gateway instances to be greater than zero for a given product, you create a dependency on PAS for that product installation. If you attempt to install a product tile with an PAS dependency before installing PAS, the install fails.

To change the number of Gateway instances, click the product tile, then select **Settings > Resource sizes > INSTANCES** and change the value next to the product Gateway job.

To remove the PAS dependency, change the value of this field to `0`.

Out of Disk Space Error

PCF displays an `Out of Disk Space` error if log files expand to fill all available disk space. If this happens, rebooting the PCF installation VM clears the tmp directory of these log files and resolves the error.

Installing Ops Manager Director Fails

If the DNS information for the PCF VM is incorrectly specified when deploying the PCF .ova file, installing Ops Manager Director fails at the “Installing Micro BOSH” step.

To resolve this issue, correct the DNS settings in the PCF Virtual Machine properties.

Deleting Ops Manager Fails

Ops Manager displays an error message when it cannot delete your installation. This scenario might happen if the Ops Manager Director cannot access the VMs or is experiencing other issues. To manually delete your installation and all VMs, you must do the following:

1. Use your IaaS dashboard to manually delete the VMs for all installed products, with the exception of the Ops Manager VM.
2. SSH into your Ops Manager VM and remove the `installation.yml` file from `/var/tempest/workspaces/default/`.

> **Note**: Deleting the `installation.yml` file does not prevent you from reinstalling Ops Manager. For future deploys, Ops Manager regenerates this file when you click **Save** on any page in the Ops Manager Director.

Your installation is now deleted.

Installing PAS Fails

If the DNS information for the PCF VM becomes incorrect after Ops Manager Director has been installed, installing PAS with Pivotal Operations Manager fails at the “Verifying app push” step.

To resolve this issue, correct the DNS settings in the PCF Virtual Machine properties.

Ops Manager Hangs During MicroBOSH Install or HAProxy States “IP Address Already Taken”

During an Ops Manager installation, you might receive the following errors:

- The Ops Manager GUI shows that the installation stops at the “Setting MicroBOSH deployment manifest” task.
- When you set the IP address for the HAProxy, the “IP Address Already Taken” message appears.

When you install Ops Manager, you assign it an IP address. Ops Manager then takes the next two consecutive IP addresses, assigns the first to MicroBOSH, and reserves the second. For example:

```
203.0.113.1 - Ops Manager (User assigned)
203.0.113.2 - MicroBOSH (Ops Manager assigned)
203.0.113.3 - Reserved (Ops Manager reserved)
```

To resolve this issue, ensure that the next two subsequent IP addresses from the manually assigned address are unassigned.

Poor PCF Performance

If you notice poor network performance by your PCF deployment and your deployment uses a Network Address Translation (NAT) gateway, your NAT gateway may be under-resourced.

Troubleshoot

To troubleshoot the issue, set a custom firewall rule in your IaaS console to route traffic originating from your private network directly to an S3-compatible object store. If you see decreased average latency and improved network performance, perform the solution below to scale up your NAT gateway.

Scale Up Your NAT Gateway

Perform the following steps to scale up your NAT gateway:

1. Navigate to your IaaS console.
2. Spin up a new NAT gateway of a larger VM size than your previous NAT gateway.
3. Change the routes to direct traffic through the new NAT gateway.
4. Spin down the old NAT gateway.

The specific procedures will vary depending on your IaaS. Consult your IaaS documentation for more information.

Common Issues Caused by Firewalls

This section describes various issues you might encounter when installing PAS in an environment that uses a strong firewall.

DNS Resolution Fails

When you install PCF in an environment that uses a strong firewall, the firewall might block DNS resolution. To resolve this issue, refer to the Troubleshooting DNS Resolution Issues section of the Preparing Your Firewall for Deploying PCF topic.

Advanced Troubleshooting with the BOSH CLI

Page last updated:

This topic describes using the BOSH CLI to help diagnose and resolve issues with your Pivotal Cloud Foundry (PCF) deployment. Before using the information and techniques in this topic, review Diagnosing Problems in PCF.

To follow the steps in this topic, you must log in to the BOSH Director. The BOSH Director runs on the virtual machine (VM) that Ops Manager deploys on the first install of the Ops Manager Director tile.

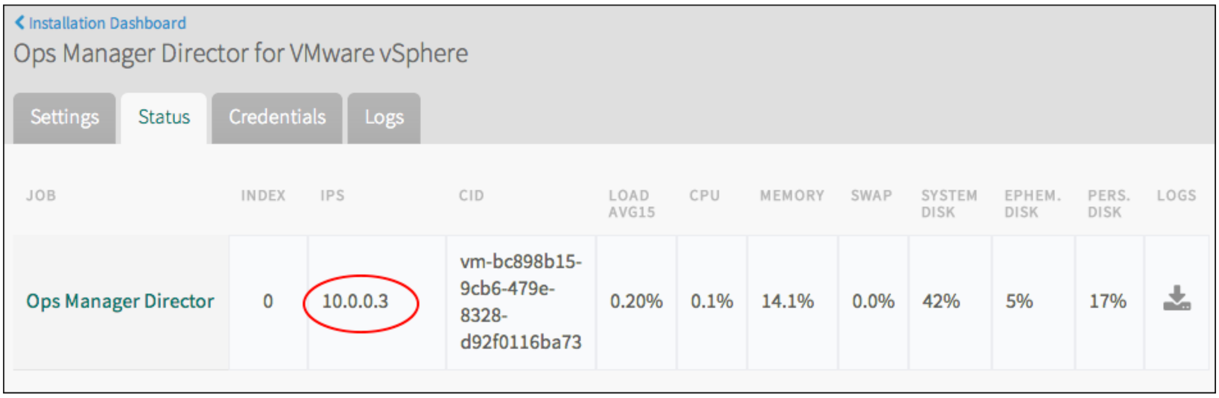
After authenticating into the BOSH Director, you can run specific commands using the BOSH Command Line Interface (BOSH CLI). BOSH Director diagnostic commands have access to information about your entire Pivotal Cloud Foundry (PCF) installation.

Note: Before running any BOSH CLI commands, verify that no BOSH Director tasks are running on the Ops Manager VM. See the *Tasks* section of BOSH CLI commands for more information.

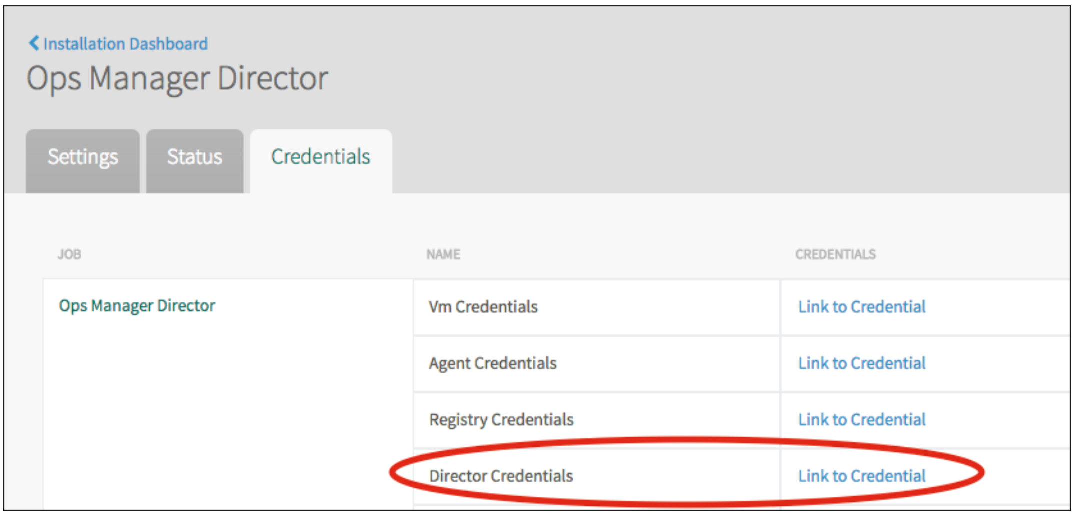
Gather Credential and IP Address Information

Before you begin troubleshooting with the BOSH CLI, follow the instructions below to collect the information you need from the Ops Manager interface.

1. Open the Ops Manager interface by navigating to the Ops Manager fully qualified domain name (FQDN) in a web browser.
2. Click the **Ops Manager Director** tile and select the **Status** tab.
3. Record the IP address for the Director job. This is the IP address of the VM where the BOSH Director runs.



4. Select the **Credentials** tab.
5. Click **Link to Credential** to view the **Director Credentials**. Record these credentials.



6. Return to the **Installation Dashboard**.

7. **(Optional)** To prepare to troubleshoot the job VM for any other product, click the product tile and repeat the procedure above to record the IP address and VM credentials for that job VM.

8. Log out of Ops Manager.

> **Note**: Ensure that there are no Ops Manager installations or updates in progress while using the BOSH CLI.

SSH into Ops Manager

Use SSH to connect to the Ops Manager web application VM. Follow the instructions in one of the sections below to SSH into the Ops Manager VM.

AWS

To SSH into the Ops Manager VM in AWS, you need the key pair you used when you created the Ops Manager VM. To see the name of the key pair, click on the Ops Manager VM and locate the `key pair name` in the properties.

Perform the following steps to SSH into the Ops Manager VM in AWS:

1. Locate the Ops Manager FQDN on the AWS **EC2 instances** page.

2. Run `chmod 600 ops_mgr.pem` to change the permissions on the `.pem` file to be more restrictive. For example:

```
$ chmod 600 ops_mgr.pem
```

3. Run `ssh -i ops_mgr.pem ubuntu@OPS-MANAGER-FQDN` to SSH into the Ops Manager VM. Replace `OPS-MANAGER-FQDN` with the fully qualified domain name of Ops Manager. For example:

```
$ ssh -i ops_mgr.pem ubuntu@my-opsmanager-fqdn.example.com
```

Azure

To SSH into the Ops Manager VM in Azure, you need the key pair you used when creating the Ops Manager VM. If you need to reset the SSH key, locate the Ops Manager VM in the Azure portal and click **Reset Password**.

Perform the following steps to SSH into the Ops Manager VM in Azure:

1. Locate the Ops Manager FQDN by selecting the VM in the Azure portal.

2. Run `chmod 600 ops_mgr.pem` to change the permissions on the `.pem` file to be more restrictive. For example:

```
$ chmod 600 ops_mgr.pem
```

3. Run `ssh -i ops_mgr.pem ubuntu@OPS-MANAGER-FQDN` to SSH into the Ops Manager VM. Replace `OPS-MANAGER-FQDN` with the fully qualified domain name of Ops Manager. For example:

```
$ ssh -i ops_mgr.pem ubuntu@my-opsmanager-fqdn.example.com
```

OpenStack

To SSH into the Ops Manager VM in OpenStack, you need the key pair that you created in the Configure Security step of the *Provisioning the OpenStack Infrastructure* topic. If you need to reset the SSH key, locate the Ops Manager VM in the OpenStack console and boot it in recovery mode to generate a new key pair.

Perform the following steps to SSH into the Ops Manager VM in OpenStack:

1. Locate the Ops Manager FQDN on the **Access & Security** page.

2. Run `chmod 600 ops_mgr.pem` to change the permissions on the `.pem` file to be more restrictive. For example:

```
$ chmod 600 ops_mgr.pem
```

3. Run `ssh -i ops_mgr.pem ubuntu@OPS-MANAGER-FQDN` to SSH into the Ops Manager VM. Replace `OPS-MANAGER-FQDN` with the fully qualified domain name of Ops Manager. For example:

```
$ ssh -i ops_mgr.pem ubuntu@my-opsmanager-fqdn.example.com
```

GCP

To SSH into the Ops Manager VM in GCP, follow these instructions:

1. Confirm that you have installed the gcloud CLI. See the Google Cloud Platform documentation for more information.
2. From the GCP console, click **Compute Engine**.
3. Locate the Ops Manager VM in the **VM Instances** list.
4. Click the **SSH** menu button.
5. Copy the SSH command that appears in the popup window.
6. Paste the command into your terminal window to SSH to the Ops Manager VM. For example:

```
$ gcloud compute ssh om-pcf-1a --zone us-central1-b
```

7. Run `sudo su - ubuntu` to switch to the `ubuntu` user.

vSphere

To SSH into the Ops Manager VM in vSphere, you need the credentials used to import the PCF .ova or .ovf file into your virtualization system. You set these credentials when you installed Ops Manager.

Note: If you lose your credentials, you must shut down the Ops Manager VM in the vSphere UI and reset the password. See the vSphere documentation for more information.

1. From a command line, run `ssh ubuntu@OPS-MANAGER-FQDN` to SSH into the Ops Manager VM. Replace `OPS-MANAGER-FQDN` with the fully qualified domain name of Ops Manager.
2. When prompted, enter the password that you set during the .ova deployment into vCenter. For example:

```
$ ssh ubuntu@my-opsmanager-fqdn.example.com
Password: ***********
```

Log in to the BOSH Director

Follow the steps below to log in to the BOSH Director.

Create a Local BOSH Director Alias

1. Run the following command to create a local alias for the BOSH Director using the BOSH CLI: `bosh alias-env MY-ENV -e DIRECTOR-IP-ADDRESS --ca-cert /var/tempest/workspaces/default/root_ca_certificate`

   Replace the placeholder text with the following:
   - `MY-ENV` : Enter an alias for the BOSH Director, such as `gcp` .
   - `DIRECTOR-IP-ADDRESS` : Enter the IP address of your Ops Manager Director VM. For example:

```
$ bosh alias-env gcp -e 10.0.0.3 --ca-cert /var/tempest/workspaces/default/root_ca_certificate
```

2. Log in to the BOSH Director using one of the following options:
   - Internal User Store Login through UAA: Log in to the Director using BOSH.
   - External User Store Login through SAML: Use an external user store to log in to the BOSH Director.

Log in to the BOSH Director with UAA

1. Retrieve the Director password from the **Ops Manager Director > Credentials** tab. Alternatively, launch a browser and visit `https://OPS-MANAGER-FQDN/api/v0/deployed/director/credentials/director_credentials` to obtain the password. Replace `OPS-MANAGER-FQDN` with the fully qualified domain name of Ops Manager.

2. Run `bosh -e MY-ENV log-in` to log in to the BOSH Director. Replace `MY-ENV` with the alias for your BOSH Director. For example:

```
$ bosh -e gcp log-in
```

   Follow the BOSH CLI prompts and enter the Ops Manager Director credentials to log in to the BOSH Director.

Log in to the BOSH Director with SAML

1. Log in to your identity provider and use the following information to configure SAML Service Provider Properties:
   - **Service Provider Entity ID**: `bosh-uaa`
   - **ACS URL**: `https://DIRECTOR-IP-ADDRESS:8443/saml/SSO/alias/bosh-uaa`
   - **Binding**: HTTP Post
   - **SLO URL**: `https://DIRECTOR-IP-ADDRESS:8443/saml/SSO/alias/bosh-uaa`
   - **Binding**: HTTP Redirect
   - **Name ID**: Email Address

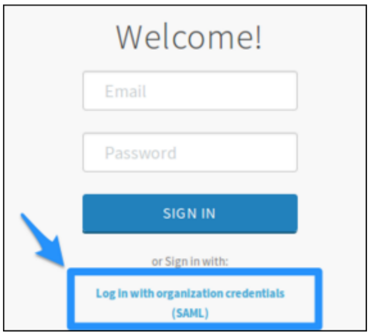
2. Run `bosh -e MY-ENV log-in` to log in to the BOSH Director. Replace `MY-ENV` with the alias for your BOSH Director. For example:

```
$ bosh -e gcp log-in
```

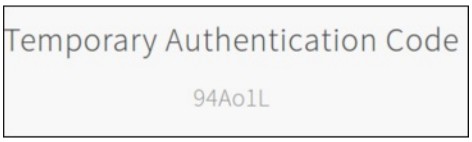
   Follow the BOSH CLI prompts and enter your SAML credentials to log in to the BOSH Director.

   Note: Your browser must be able to reach the BOSH Director in order to log in with SAML.

3. Click **Log in with organization credentials (SAML)**.



4. Copy the **Temporary Authentication Code** that appears in your browser.



5. You see a login confirmation. For example:

```
Logged in as admin@example.org
```

Use the BOSH CLI for Troubleshooting

This section describes three BOSH CLI commands commonly used during troubleshooting.

- **VMs**: Lists the VMs in a deployment
- **Cloud Check**: Runs a cloud consistency check and interactive repair
- **SSH**: Starts an interactive session or executes commands with a VM

BOSH VMs

The `bosh vms` command provides an overview of the virtual machines that BOSH manages.

To use this command, run `bosh -e MY-ENV vms` to see an overview of all virtual machines managed by BOSH, or `bosh -e MY-ENV -d MY-DEPLOYMENT vms` to see only the virtual machines associated with a particular deployment. Replace `MY-ENV` with your environment, and, if using the `-d` flag, also replace `MY-DEPLOYMENT` with the name of a deployment.

When troubleshooting an issue with your deployment, `bosh vms` may show a VM in an **unknown** state. Run bosh cloud-check on a VM in an **unknown** state to instruct BOSH to diagnose problems with the VM.

You can also run `bosh vms` to identify VMs in your deployment, then use the bosh ssh command to SSH into an identified VM for further troubleshooting.

`bosh vms` supports the following arguments:

- `--dns`: Report also includes the DNS A record for each VM
- `--vitals`: Report also includes load, CPU, memory usage, swap usage, system disk usage, ephemeral disk usage, and persistent disk usage for each VM

Note: The **Status** tab of the Pivotal Application Service (PAS) product tile displays information similar to the `bosh vms` output.

BOSH Cloud Check

Run the `bosh cloud-check` command to instruct BOSH to detect differences between the VM state database maintained by the BOSH Director and the actual state of the VMs. For each difference detected, `bosh cloud-check` can offer the following repair options:

- `Reboot VM`: Instructs BOSH to reboot a VM. Rebooting can resolve many transient errors.
- `Ignore problem`: Instructs BOSH to do nothing. You may want to ignore a problem in order to run `bosh ssh` and attempt troubleshooting directly on the machine.
- `Reassociate VM with corresponding instance`: Updates the BOSH Director state database. Use this option if you believe that the BOSH Director state database is in error and that a VM is correctly associated with a job.
- `Recreate VM using last known apply spec`: Instructs BOSH to destroy the server and recreate it from the deployment manifest that the installer provides. Use this option if a VM is corrupted.
- `Delete VM reference`: Instructs BOSH to delete a VM reference in the Director state database. If a VM reference exists in the state database, BOSH expects to find an agent running on the VM. Select this option only if you know that this reference is in error. Once you delete the VM reference, BOSH can no longer control the VM.

To use this command, run `bosh -e MY-ENV -d MY-DEPLOYMENT cloud-check`. Replace `MY-ENV` with your environment, and `MY-DEPLOYMENT` with your deployment.

Example Scenarios

Unresponsive Agent

```
$ bosh -e example-env -d example-deployment cloud-check
ccdb/0 (vm-3e37133c-bc33-450e-98b1-f86d5b63502a) is not responding:

- Ignore problem
- Reboot VM
- Recreate VM using last known apply spec
- Delete VM reference (DANGEROUS!)
```

Missing VM

```
$ bosh -e example-env -d example-deployment cloud-check
VM with cloud ID `vm-3e37133c-bc33-450e-98b1-f86d5b63502a' missing:

- Ignore problem
- Recreate VM using last known apply spec
- Delete VM reference (DANGEROUS!)
```

Unbound Instance VM

```
$ bosh -e example-env -d example-deployment cloud-check
VM `vm-3e37133c-bc33-450e-98b1-f86d5b63502a' reports itself as `ccdb/0' but does not have a bound instance:

- Ignore problem
- Delete VM (unless it has persistent disk)
- Reassociate VM with corresponding instance
```

Out of Sync VM

```
$ bosh -e example-env -d example-deployment cloud-check
VM `vm-3e37133c-bc33-450e-98b1-f86d5b63502a' is out of sync:
expected `cf-d7293430724a2c421061: ccdb/0', got `cf-d7293430724a2c421061: nats/0':

- Ignore problem
- Delete VM (unless it has persistent disk)
```

BOSH SSH

Use `bosh ssh` to SSH into the VMs in your deployment.

Follow the steps below to use `bosh ssh` :

1. Identify a VM to SSH into. Run `bosh -e MY-ENV -d MY-DEPLOYMENT vms` to list the VMs in the given deployment. Replace `MY-ENV` with your environment alias and `MY-DEPLOYMENT` with the deployment name.

2. Run `bosh -e MY-ENV -d MY-DEPLOYMENT ssh VM-NAME/GUID` . For example:

   ```
   $ bosh -e example-env -d example-deployment ssh diego-cell/abcd0123-a012-b345-c678-9def01234567
   ```

Pivotal

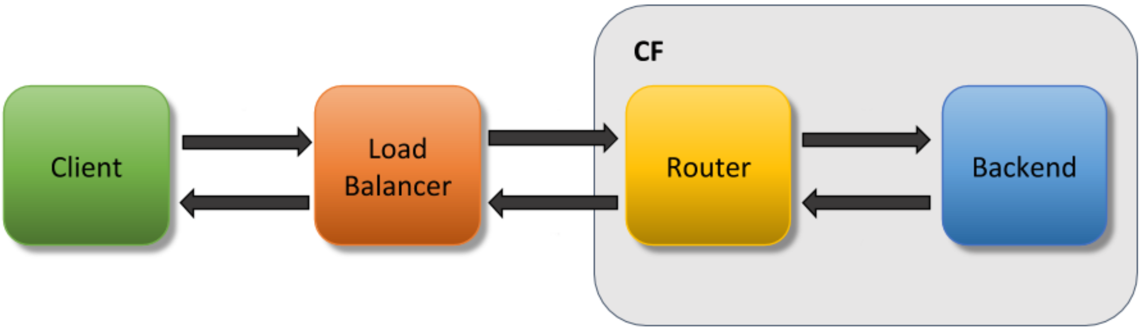
Troubleshooting Slow Requests in Cloud Foundry

Page last updated:

This topic suggests ways that an operator of Cloud Foundry (CF) can diagnose the location of app request delays.

App Request Path

App requests typically transit the following components. Only the router (Gorouter) and app are within the scope of Cloud Foundry. Operators may have the HAProxy load balancer that comes with CF deployed, instead of or in addition to, an infrastructure load balancer.



You can use `time` to measure a request's full round trip time from the client and back, examine `cf logs` output to measure the time just within Cloud Foundry, and add log messages to your app for fine-grained measurements of where the app itself takes time. By comparing these times, you can determine whether your delay comes from outside CF, inside the Gorouter, or in the app.

The following sections describe a scenario of diagnosing the source of delay for an app `app1`.

Measure Total Round-Trip App Requests

On a command line, run `time curl -v APP-ENDPOINT` to measure the total round-trip time for deployed app `app1`. For example:

```
$ time curl -v http://app1.app_domain.com

GET /hello HTTP/1.1
Host: app1.app_domain.com
User-Agent: curl/7.43.0
Accept: */*

HTTP/1.1 200 OK
Content-Type: application/json;charset=utf-8
Date: Tue, 14 Dec 2016 00:31:32 GMT
Server: nginx
X-Content-Type-Options: nosniff
X-Vcap-Request-Id: c30fad28-4972-46eb-7da6-9d07dc79b109
Content-Length: 602
hello world!
real	2m0.707s
user	0m0.005s
sys	0m0.007s
```

The `real` time output shows that the request to `http://app1.app_domain.com` took approximately 2 minutes, round-trip. This seems like an unreasonably long time, so it makes sense to find out where the delay is occurring. To narrow it down, the next step measures the part of that request response time that comes from within Cloud Foundry.

Note: If your `curl` outputs an error like `Could not resolve host: NONEXISTENT.com` then DNS failed to resolve. If `curl` returns normally but lacks a `X-Vcap-Request-Id`, the request from the Load Balancer did not reach Cloud Foundry.

Measure App Requests within Cloud Foundry

The `cf logs` command streams log messages from the Gorouter as well as from apps. To see the timestamps of Gorouter messages to and from your app, do the following:

1. If necessary, run `cf apps` to determine the name of the app.
2. Run `cf logs APP-NAME`. Replace `APP-NAME` with the name of the app.
3. From another terminal window, send a request to your app.
4. After your app returns a response, enter `Ctrl-C` to stop streaming `cf logs`.

For example:

```
$ cf logs app1

2016-12-14T00:33:32.35-0800 [RTR/0]      OUT app1.app_domain.com - [14/12/2016:00:31:32.348 +0000] "GET /hello HTTP/1.1" 200 0 60 "-" "HTTPClient/1.0 (2.7.1, ruby 2.3.3 (2016-11-21
2016-12-14T00:32:32.35-0800 [APP/PROC/WEB/0]OUT app1 received request at [14/12/2016:00:32:32.348 +0000] with "vcap_request_id": "01144146-1e7a-4c77-77ab-49ae3e286fe9"
^C
```

In the example above, the first line contains timestamps from the Gorouter for both when it received the request and what was its response time processing the request:

- `14/12/2016:00:31:32.348` : Gorouter receives request
- `response_time:120.00641734` : Gorouter round-trip processing time

This output shows that it took 120 seconds for the Gorouter to process the request, which means that the 2-minute delay above takes place within CF, either within the Gorouter or within the app.

To determine whether the app is responsible, add logging to your app to measure where it is spending time.

> **Note**: Every incoming request should generate an access log message. If a request does not generate an access log message, it means the Gorouter did not receive the request.

Use App Logs to Locate Delays in CF

To gain a more detailed picture of where delays exist in your request path, augment the logging that your app generates. For example, call your logging library from the request handler to generate log lines when your app receives a request and finishes processing it:

```
2016-12-14T00:33:32.35-0800 [RTR/0]      OUT app1.app_domain.com - [14/12/2016:00:31:32.348 +0000] "GET /hello HTTP/1.1" 200 0 60 "-" "HTTPClient/1.0 (2.7.1, ruby 2.3.3 (2016-11-21
2016-12-14T00:32:32.35-0800 [APP/PROC/WEB/0]OUT app1 received request at [14/12/2016:00:32:32.348 +0000] with "vcap_request_id": "01144146-1e7a-4c77-77ab-49ae3e286fe9"
2016-12-14T00:32:32.50-0800 [APP/PROC/WEB/0]OUT app1 finished processing req at [14/12/2016:00:32:32.500 +0000] with "vcap_request_id": "01144146-1e7a-4c77-77ab-49ae3e286fe9"
```

Comparing the router access log messages from the previous section with the new app logs above, we can construct the following timeline:

- `14/12/2016:00:31:32.348` : Gorouter receives request
- `2016-12-14T00:32:32.35` : App receives request
- `2016-12-14T00:32:32.50` : App finishes processing request
- `2016-12-14T00:33:32.35` : Gorouter finishes processing request

The timeline indicates that the Gorouter took close to 60 seconds to send the request to the app and another 60 seconds to receive the response from the app. This suggests a delay either with the Gorouter, or in network latency between the Gorouter and Diego cells hosting the app.

Time the Gorouter Processing

To determine whether a Gorouter delay comes from the Gorouter itself or network latency between the Gorouter and the app, log into the Gorouter and compare the response times from calling the app two different ways:

- Call the app through the router proxy to process requests through the Gorouter
- Directly call a specific instance of the app, bypassing Gorouter processing

To make this comparison, do the following:

1. Log in to the `router` using `bosh ssh`. See the BOSH SSH documentation for more information.

2. Use `time` and `curl` to measure the response time of an app request originating from and processing through the Gorouter, but not running through the client network or load balancer:

```
$ time curl -H "Host: app1.app_domain.com" http://IP-GOROUTER-VM:80"
```

3. Obtain the IP address and port of a specific app instance by running the following and recording associated `host` and `port` values:

```
$ cf curl /v2/apps/$(cf app app1 --guid)/stats
{
  "0": {
  "state": "RUNNING",
  "stats": {
    [...]
    "host": "10.10.148.39",
    "port": 60052,
    [...]
  },
[...]
}
```

4. Use the IP address and port values to measure response time calling the app instance directly, bypassing Gorouter processing:

```
$ time curl http://APP-HOST-IP:APP-PORT
```

If the Gorouter and direct response times are similar, it suggests network latency between the Gorouter and the Diego cell. If the Gorouter time is much longer than the direct-to-instance time, the Gorouter is slow processing requests. The next section explains why the Gorouter might be slow.

Potential Causes for Gorouter Latency

- Routers are under heavy load from incoming client requests.
- Apps are taking a long time to process requests. This increases the number of concurrent threads held open by the Gorouter, reducing capacity to handle requests for other apps.

Operations Recommendations

- Monitor CPU load for Gorouters. At high CPU (70%+), latency increases. If the Gorouter CPU reaches this threshold, consider adding another Gorouter instance.
- Monitor latency of all routers using metrics from the Firehose. Do not monitor the average latency across all routers. Instead, monitor them individually on the same graph.
- Consider using Pingdom against an app on your Cloud Foundry deployment to monitor latency and uptime.
- Consider enabling access logs on your load balancer. See your load balancer documentation for how. Just as we used Gorouter access log messages above to determine latency from the Gorouter, you can compare load balancer logs to identify latency between the load balancer and the Gorouter. You can also compare load balancer response times with the client response times to identify latency between client and load balancer.
- Deploy a nozzle to the Loggregator Firehose to track metrics for the Gorouter. Available metrics include:
  - CPU utilization
  - Latency
  - Requests per second

Troubleshooting TCP Routes

Page last updated:

The following topic provides steps to determine whether TCP routing issues are related to DNS and load balancer misconfiguration, the TCP routing tier, or the routing subsystem.

Rule Out the App

If you are having TCP routing issues with an app, follow the procedure below to determine what to troubleshoot.

Prerequisites

This procedure requires that you have the following:

- An app with TCP routing issues.
- A TCP domain. See Routes and Domains for more information about creating a TCP domain.
- A simple HTTP web app that you can use to curl.

Procedure

1. Push a simple HTTP app using your TCP domain by entering the following command:

```
$ cf push MY-APP -d tcp.MY-DOMAIN --random-route
```

2. Curl your app on the port generated for the route. For example, if the port is 1024:

```
$ curl tcp.MY-DOMAIN:1024
```

3. If the curl request fails to reach the app, proceed to the next section: Rule Out DNS and the Load Balancer.
4. If the curl request to your simple app succeeds, curl the app you are having issues with.
5. If you cannot successfully curl your problem app, TCP routing is working correctly. There is an issue with the app you cannot successfully curl.

Rule Out DNS and the Load Balancer

1. Curl the TCP router healthcheck endpoint:

```
$ curl tcp.MY-DOMAIN:80/health -v
```

2. If you receive a `200 OK` response, proceed to the next section: Rule Out the Routing Subsystem.
3. If you do not receive a `200 OK`, your load balancer may not be configured to pass through the healthcheck port. Continue following this procedure to test your load balancer configuration.
4. Confirm that your TCP domain name resolves to your load balancer:

```
$ dig tcp.MY-DOMAIN
...
tcp.MY-DOMAIN. 300 IN  A 123.456.789.123
```

5. As an admin user, list the reservable ports configured for the `default-tcp` router group:

```
$ cf curl /routing/v1/router_groups
[
  {
    "guid": "d9b1db52-ea78-4bb9-7473-ec8e5d411b14",
    "name": "default-tcp",
    "type": "tcp",
    "reservable_ports": "1024-1123"
  }
]
```

6. Choose a port from the `reservable_ports` range and curl the TCP domain on this port. For example, if you chose port 1024:

```
$ curl tcp.MY-DOMAIN:1024 -v
```

7. If you receive an immediate rejection, then the TCP router likely rejected the request because there is no route for this port.

8. If your connection times out, then you need to configure your load balancer to route all ports in `reservable_ports` to the TCP routers.

Rule Out the Routing Subsystem

Send a direct request to the TCP router to confirm that it routes to your app.

1. SSH into your TCP router.

2. Curl the port of your route using the IP address of the router itself. For example, if the port reserved for your route is `1024`, and the IP address of the TCP router is `10.0.16.18`:

```
$ curl 10.0.16.18:1024
```

3. If the curl is successful, then the load balancer either:
    - cannot reach the TCP routers, or
    - is not configured to route requests to the TCP routers from the `reservable_ports`

4. If you cannot reach the app by curling the TCP router directly, perform the following steps to confirm that your TCP route is in the routing table.

    a. Record the **Tcp Emitter Credentials** from the **UAA** row in the PAS **Credentials** tab. This OAuth client has permissions to read the routing table.

    b. Install the UAA CLI `uaac`:

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

    c. Obtain a token for this OAuth client from UAA by providing the client secret:

    ```
    $ uaac token client get tcp_emitter
    Client secret:
    ```

    d. Obtain an access_token:

    ```
    $ uaac context
    ```

    e. Use the routing API to list TCP routes:

    ```
    $ curl api.MY-DOMAIN/routing/v1/tcp_routes -H "Authorization: bearer TOKEN"
    [{"router_group_guid":"f3518f7d-d8a0-4279-4e89-c058040d0000",
    "backend_port":60000,"backend_ip":"10.244.00.0","port":60000,"modification_tag":{"guid":"d4cc3bbe-c838-4857-7360-19f034440000",
    "index":1},"ttl":120}]
    ```

    f. In this output, each route mapping has the following:
        - **port**: your route port
        - **backend_ip**: an app instance mapped to the route
        - **backend_port**: the port number for the app instance mapped to the route

    g. If your route port is not in the response, then the `tcp_emitter` may be unable to register TCP routes with the routing API. Look at the logs for `tcp_emitter` to see if there are any errors or failures.

    h. If the route is in the response, but you were not able to curl the port on the TCP route directly, then the TCP router may be unable to reach the routing API. Look at the logs for `tcp_router` to see if there are any errors or failures.

Recovering MySQL from PAS Downtime

Page last updated:

This topic assumes you are using BOSH CLI v2.

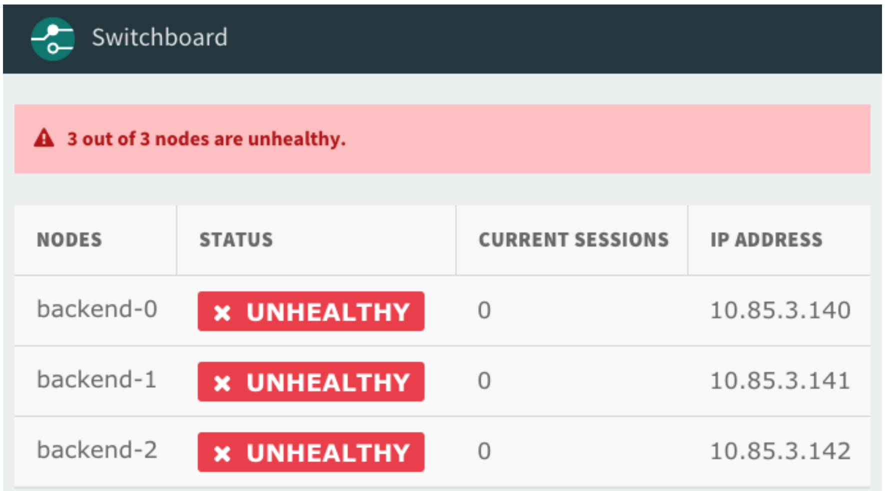
This topic describes the procedure for recovering a terminated Pivotal Application Service (PAS) cluster using a process known as bootstrapping.

When to Bootstrap

You must bootstrap a cluster that loses quorum. A cluster loses quorum when less than half of the nodes can communicate with each other for longer than the configured grace period. If a cluster does not lose quorum, individual unhealthy nodes automatically rejoin the cluster after resolving the error, restarting the node, or restoring connectivity.

You can detect lost quorum through the following symptoms:

- All nodes appear “Unhealthy” on the proxy dashboard, viewable at `proxy-BOSH-JOB-INDEX.p-mysql.YOUR-SYSTEM-DOMAIN`:



- All responsive nodes report the value of `wsrep_cluster_status` as `non-Primary`:

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

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

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

See the Cluster Scaling, Node Failure, and Quorum topic for more details about determining cluster state.

Follow the steps below to recover a cluster that has lost quorum.

Step 1: Choose the Correct Manifest

1. Log in to the BOSH Director by running `bosh -e MY-ENV log-in`. Replace `MY-ENV` with the environment where you deployed the cluster.

```
$ bosh -e prod log-in
```

2. Run `bosh -e MY-ENV deployments`. Replace `MY-ENV` with the environment where you deployed the cluster.

```
$ bosh -e prod deployments
Using environment '192.168.56.6' as client 'admin'
Name             Release(s)             Stemcell(s)                                  Team(s)  Cloud Config
cf               binary-buildpack/1.0.9    bosh-warden-boshlite-ubuntu-trusty-go<\_agent/3363.9  -       latest
                 capi/1.21.0
                 cf-mysql/34
                 cf-smoke-tests/11
                 cflinuxfs2-rootfs/1.52.0
                 consul/155
                 diego/1.8.1
                 garden-runc/1.2.0
                 loggregator/78
                 nats/15
                 routing/0.145.0
                 statsd-injector/1.0.20
                 uaa/25
service-instance    mysql/0.6.0            bosh-warden-boshlite-ubuntu-trusty-go\_agent/3363.9  -       latest

2 deployments

Succeeded
```

3. Run `bosh -e MY-ENV -d MY-DEPLOYMENT manifest > /tmp/MANIFEST.yml` to download the manifest. Replace the example text with the following:
   - `MY-ENV` : the environment where you deployed the cluster
   - `MY-DEPLOYMENT` : the name of your deployment cluster
   - `MANIFEST.yml` : the name you want to give the manifest

```
$ bosh -e prod -d mysql manifest /tmp/mysql.yml
```

Step 2: Run the Bootstrap Errand

PAS versions 1.7.0 and later include a BOSH errand to automate the process of bootstrapping. The bootstrap errand automates the steps described in the Manual Bootstrapping section below. It finds the node with the highest transaction sequence number and asks it to start up by itself in bootstrap mode. Finally, it asks the remaining nodes to join the cluster.

In most cases, running the errand will recover your cluster. However, certain scenarios require additional steps. To determine which set of instructions to follow, you must determine the state of your Virtual Machines (VMs).

1. Run `bosh -e MY-ENV -d MY-DEPLOYMENT instances` and examine the output. Replace `MY-ENV` with the environment where you deployed the cluster and `MY-DEPLOYMENT` with the deployment cluster name.

```
$ bosh -e prod -d mysql instances
```

   - If the output of `bosh instances` shows the state of the jobs as `failing` , proceed to Scenario 1.
   - If the output of `bosh instances` shows the state of jobs as `unknown/unknown` , proceed to Scenario 2.

Scenario 1: Virtual Machines Running, Cluster Disrupted

In this scenario, nodes are up and running, but the cluster has been disrupted. You can run the bootstrap errand without recreating the VMs.

1. Run `bosh -e MY-ENV -d MY-DEPLOYMENT run-errand bootstrap` . Replace `MY-ENV` with the name of the environment where you deployed the cluster and `MY-DEPLOYMENT` with the deployment cluster name.

```
$ bosh -e prod -d mysql run-errand  bootstrap
```

> **Note**: Sometimes the bootstrap errand fails on the first try. If this happens, run the command again in a few minutes.

2. If the errand fails, try performing the steps automated by the errand manually by following the Manual Bootstrapping procedure.

Scenario 2: Virtual Machines Terminated or Lost

In this scenario, severe circumstances such as power failure have terminated all of your VMs. You need to recreate the VMs before you can recover the cluster.

1. To recreate terminated or lost VMs, perform the following steps:

   a. If you use the VM Resurrector, disable it.
   b. Run the BOSH Cloud Check interactive command `bosh -e MY-ENV -d MY-DEPLOYMENT cloud-check`. When prompted, select **Recreate VM and wait for processes to start**. If this option fails, select **Delete VM reference**.

   ```
   $ bosh -e prod -d mysql cloud-check

   Using environment '192.168.56.6' as user 'director' (bosh.*.read, openid, bosh.*.admin, bosh.read, bosh.admin)

   Task 34

   19:19:12 | Scanning 21 VMs: Checking VM states (00:00:16)
   19:19:28 | Scanning 21 VMs: 19 OK, 2 unresponsive, 0 missing, 0 unbound (00:00:00)
   19:19:28 | Scanning 5 persistent disks: Looking for inactive disks (00:00:00)
   19:19:28 | Scanning 5 persistent disks: 5 OK, 0 missing, 0 inactive, 0 mount-info mismatch (00:00:00)

   Started  Fri Aug  4 19:19:12 UTC 2017
   Finished Fri Aug  4 19:19:28 UTC 2017
   Duration 00:00:16

   Task 34 done

   #  Type              Description
   1  unresponsive_agent  VM for 'uaa/0 (0)' with cloud ID 'vm-001' is not responding.
   2  unresponsive_agent  VM for 'mysql/0 (0)' with cloud ID 'vm-007' is not responding.

   2 problems

   1: Skip for now
   2: Reboot VM
   3: Recreate VM without waiting for processes to start
   4: Recreate VM and wait for processes to start
   5: Delete VM
   6: Delete VM reference (forceful; may need to manually delete VM from the Cloud to avoid IP conflicts)
   ```

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

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

2. Complete the following steps to prepare your deployment for the bootstrap errand:

   a. Open `/tmp/MANIFEST.yml` in a text editor.
   b. Search for the jobs section: `jobs`.
   c. Search for the mysql-partition: `mysql-partition`.
   d. Search for the update section: `update`.
   e. Change `max_in_flight` to `3`.
   f. Below the `max_in_flight` line, add a new line: `canaries: 0`.
   g. Set `update.serial` to `false`.
   h. Run `bosh -e MY-ENV -d MY-DEPLOYMENT deploy /tmp/MANIFEST.yml`.

3. Run `bosh -e MY-ENV -d MY-DEPLOYMENT run-errand bootstrap`. Replace `MY-ENV` with the name of the environment where you deployed the cluster and `MY-DEPLOYMENT` with the deployment cluster name.

4. Run `bosh -e MY-ENV -d MY-DEPLOYMENT instances` and examine the output to confirm that the errand completes successfully. Some instances may still appear as `failing`.

5. Complete the following steps to restore the BOSH configuration:

   a. Open `/tmp/MANIFEST.yml` in a text editor.
   b. Re-set `canaries` to 1, `max_in_flight` to 1, and `serial` to true in the same manner as above.
   c. Run `bosh -e MY-ENV -d MY-DEPLOYMENT deploy /tmp/MANIFEST.yml`.
   d. Validate that all mysql instances are in `running` state.

   Note: You must reset the values in the BOSH manifest to ensure successful future deployments and accurate reporting of the status of your jobs.

6. If this procedure fails, try performing the steps automated by the errand manually by following the Manual Bootstrapping procedure.

Manual Bootstrapping

Note: The following steps are prone to user error and can result in lost data if followed incorrectly. Please follow the Run the Bootstrap Errand instructions above first, and only resort to the manual process if the errand fails to repair the cluster.

If the bootstrap errand cannot recover the cluster, you need to perform the steps automated by the errand manually.

- If the output of `bosh instances` shows the state of the jobs as `failing` (Scenario 1), proceed directly to the manual steps below.
- If the output of `bosh instances` shows the state of the jobs as `unknown/unknown` , perform Steps 1-2 of Scenario 2, substitute the manual steps below for Step 3, and then perform Steps 4-5 of Scenario 2.

1. SSH to each node in the cluster and, as root, shut down the `mariadb` process.

   ```
   $ monit stop mariadb_ctrl
   ```

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

2. Choose a node to bootstrap by locating the node with the highest transaction sequence number (`seqno`). You can obtain the `seqno` of a stopped node in one of two ways:
   - If a node shut down gracefully, the `seqno` is in the Galera state file of the node.

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

   - If the node crashed or was killed, the `seqno` in the Galera state file of the node is `-1` . In this case, the `seqno` may be recoverable from the database.
     1. Run the following command to start up the database, log the recovered sequence number, and exit.

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

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

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

        If the node never connected to the cluster before crashing, it may not have a group id (`uuid` in `grastate.dat`). In this case, you cannot recover the `seqno` . Unless all nodes crashed this way, do not choose this node for bootstrapping.

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

   Note: Only perform these bootstrap commands on the node with the highest `seqno` . Otherwise, the node with the highest `seqno` will be unable to join the new cluster unless its data is abandoned. Its `mariadb` process will exit with an error. See the Cluster Scaling, Node Failure, and Quorum topic for more details about intentionally abandoning data.

4. On the bootstrap node, update the state file and restart the `mariadb` process.

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

5. Check that the `mariadb` process has started successfully.

   ```
   $ watch monit summary
   ```

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

6. Once the bootstrapped node is running, start the `mariadb` process on the remaining nodes using `monit` .

   ```
   $ monit start mariadb_ctrl
   ```

7. Verify that the new nodes have successfully joined the cluster. The following command displays the total number of nodes in the cluster:

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

8. Complete the following steps to restore the BOSH configuration:

    a. Open `/tmp/MANIFEST.yml` in a text editor.
    b. Re-set `canaries` to 1, `max_in_flight` to 1, and `serial` to true in the same manner as above.
    c. Run `bosh -e MY-ENV -d MY-DEPLOYMENT deploy /tmp/MANIFEST.yml`.
    d. Validate that all mysql instances are in `running` state.

> **Note:** You must reset the values in the BOSH manifest to ensure successful future deployments and accurate reporting of the status of your jobs.

Troubleshooting Ops Manager for VMware vSphere

Page last updated:

This guide provides help with diagnosing and resolving issues that are specific to Pivotal Cloud Foundry (PCF) deployments on VMware vSphere.

For infrastructure-agnostic troubleshooting help, refer to Diagnosing Problems in PCF.

Common Issues

The following sections list common issues you might encounter and possible resolutions.

PCF Installation Fails

If you modify the vCenter Statistics Interval Duration setting from its default setting of 5 minutes, the PCF installation might fail at the MicroBOSH deployment stage, and the logs might contain the following error message: `The specified parameter is not correct, interval`. This failure happens because Ops Manager expects a default value of 5 minutes, and the call to this method fails when the retrieved value does not match the expected default value.

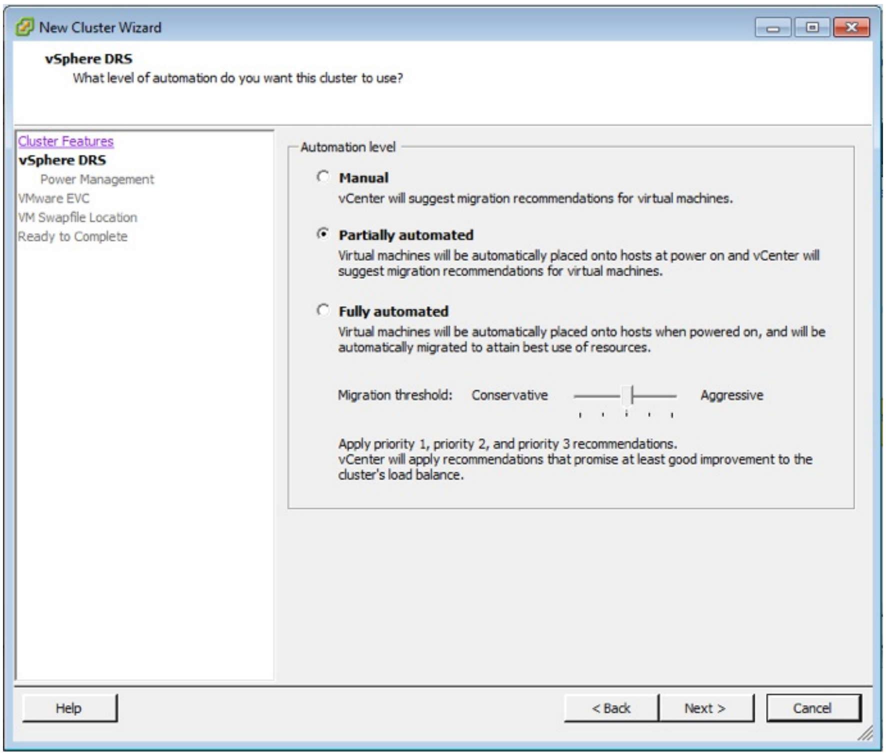
To resolve this issue, launch vCenter, navigate to **Administration > vCenter Server Settings > Statistics**, and reset the vCenter Statistics Interval Duration setting to 5 minutes.

BOSH Automated Installation Fails

Before starting an Pivotal Application Service (PAS) deployment, you must set up and configure a vSphere cluster.

If you enable vSphere DRS (Distributed Resource Scheduler) for the cluster, you must set the Automation level to **Partially automated** or **Fully automated**.

If you set the Automation level to **Manual**, the BOSH automated installation will fail with a `power_on_vm` error when BOSH attempts to create virtual VMs.



Ops Manager Loses Its IP Address After HA or Reboot

Ops Manager can lose its IP address and use DHCP due to an issue in the open source version of VMware Tools. Review the support topic for this issue in order to troubleshoot this problem.

Cannot Connect to the OVF Via a Browser

If you deployed the OVF file but cannot connect to it via a browser, check that the network settings you entered in the wizard are correct.

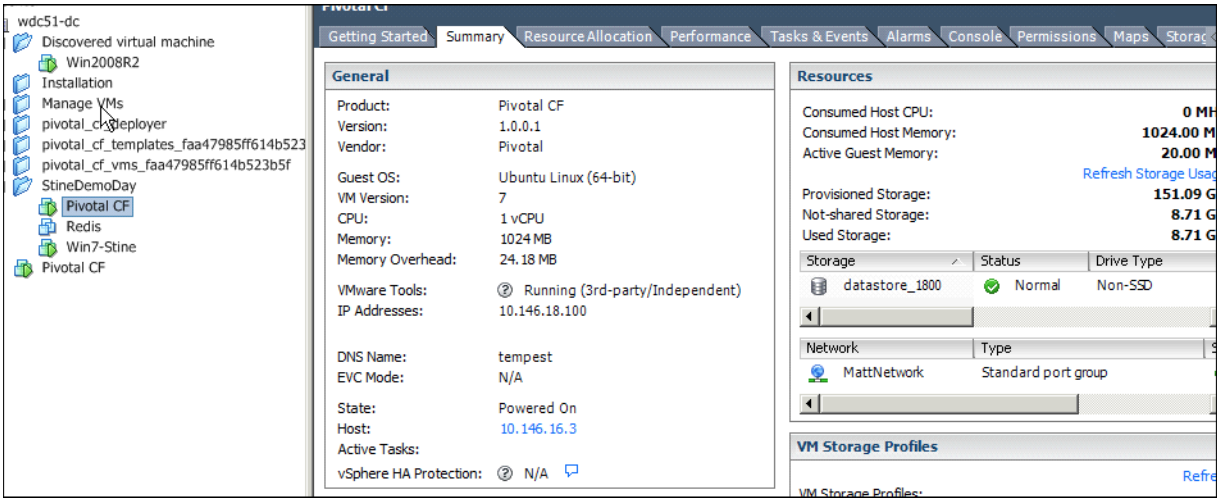
1. Access the PCF installation VM using the vSphere Console. If your network settings are misconfigured, you will not be able to SSH into the installation VM.

2. Log in using the credentials you provided when you imported the PCF .ova in vCenter.

3. Confirm that the network settings are correct by checking that the ADDRESS, NETMASK, GATEWAY, and DNS-NAMESERVERS entries are correct in `/etc/network/interfaces`.

4. If any of the settings are wrong, run `sudo vi /etc/network/interfaces` and correct the wrong entries.

5. In vSphere, navigate to the **Summary** tab for the VM and confirm that the network name is correct.



6. If the network name is wrong, right click on the VM, select **Edit Settings > Network adapter 1**, and select the correct network.

7. Reboot the installation VM.

Installation Fails with Failed Network Connection

If you experience a communication error while installing Ops Manager or MicroBOSH Director, check the following settings.

- Ensure that the routes are not blocked. vSphere environments use NSX for firewall, NAT/SNAT translation and load balancing. All communication between PCF VMs and vCenter or ESXi hosts route through the NSX firewall and are blocked by default.
- Open port 443. Ops Manager and MicroBOSH Director VMs require access to vCenter and all ESX through port 443.
- Allocate more IP addresses. BOSH requires that you allocate a sufficient number of additional dynamic IP addresses when configuring a reserved IP address range during installation. BOSH uses these IP addresses during installation to compile and deploy VMs, install PAS, and connect to services. We recommend that you allocate at least 36 dynamic IP addresses when deploying Ops Manager and PAS.

