

Table of Contents

Table of Contents	1
Installing Pivotal Cloud Foundry	7
Preparing Your Firewall for Deploying Pivotal Cloud Foundry	8
Pivotal Cloud Foundry IaaS User Role Guidelines	10
Installing Pivotal Cloud Foundry on AWS	11
Deploying the CloudFormation Template for Pivotal Cloud Foundry on AWS	13
Launching an Ops Manager Director Instance on AWS	19
Configuring Ops Manager Director on AWS	24
Deploying Elastic Runtime on AWS	38
Deleting an AWS Installation from the Console	69
Guidelines for Creating User Roles on AWS	73
Configuring Amazon EBS Encryption	74
Creating a Proxy ELB for Diego SSH without CloudFormation	77
Installing Pivotal Cloud Foundry on Azure	80
Preparing to Deploy PCF on Azure	82
Launching an Ops Manager Director Instance with an ARM Template	86
Launching an Ops Manager Director Instance on Azure without an ARM Template	90
Configuring Ops Manager Director on Azure	96
Deploying Elastic Runtime on Azure	106
Deploying PCF on Azure Government Cloud	133
Troubleshooting PCF on Azure	136
Deleting a PCF on Azure Installation	138
Installing Pivotal Cloud Foundry on GCP	139
Recommended GCP Quotas	141
Preparing to Deploy PCF on GCP	142
Launching an Ops Manager Director Instance on GCP	159
Configuring Ops Manager Director on GCP	164
Deploying Elastic Runtime on GCP	176
Deleting a GCP Installation from the Console	206
Troubleshooting PCF on GCP	208
Installing Pivotal Cloud Foundry on OpenStack	210
Provisioning the OpenStack Infrastructure	213
Configuring Ops Manager Director for OpenStack	220
Installing Elastic Runtime after Deploying Pivotal Cloud Foundry on OpenStack	235
Installing Pivotal Cloud Foundry on vSphere	264
Deploying Operations Manager to vSphere	267
Configuring Ops Manager Director for VMware vSphere	270
Configuring Elastic Runtime for vSphere	283
Provisioning a Virtual Disk in vSphere	312
Using the Cisco Nexus 1000v Switch with Ops Manager	314
Using Ops Manager Resurrector on VMware vSphere	317
Configuring Pivotal Cloud Foundry SSL Termination for vSphere Deployments	319
Understanding Availability Zones in VMware Installations	321
Upgrading Pivotal Cloud Foundry	323
Understanding the Effects of Single Components on a Pivotal Cloud Foundry Upgrade	330
Upgrade Considerations for Selecting File Storage in Pivotal Cloud Foundry	333
Upgrading Elastic Runtime and Other Pivotal Cloud Foundry Products	335
Reference Architectures	337

Reference Architecture for Pivotal Cloud Foundry on AWS	338
Reference Architecture for Pivotal Cloud Foundry on Azure	341
Reference Architecture for Pivotal Cloud Foundry on GCP	346
Reference Architectures for Pivotal Cloud Foundry on vSphere	350
PCF Dev Overview	358
Using Ops Manager	360
Using the Ops Manager API	362
Understanding the Ops Manager Interface	364
Adding and Deleting Products	366
Understanding Floating Stemcells	370
Creating UAA Clients for BOSH Director	371
Using Your Own Load Balancer	373
Understanding Pivotal Cloud Foundry User Types	375
Starting and Stopping Pivotal Cloud Foundry Virtual Machines	378
Creating and Managing Ops Manager User Accounts	382
Creating New Elastic Runtime User Accounts	384
Logging in to Apps Manager	385
Adding Existing SAML or LDAP Users to a Pivotal Cloud Foundry Deployment	386
Modifying Your Ops Manager Installation and Product Template Files	388
Managing Errands in Ops Manager	391
Backing Up and Restoring Pivotal Cloud Foundry	394
Backing Up Pivotal Cloud Foundry	395
Restoring Pivotal Cloud Foundry from Backup	403
Monitoring Virtual Machines in Pivotal Cloud Foundry	412
Pivotal Cloud Foundry Troubleshooting Guide	414
Troubleshooting Ops Manager for VMware vSphere	422
Recovering MySQL from Elastic Runtime Downtime	424
Advanced Troubleshooting with the BOSH CLI	430
Cloud Foundry Concepts	437
Cloud Foundry Overview	438
How Applications Are Staged	441
High Availability in Cloud Foundry	443
Orgs, Spaces, Roles, and Permissions	447
Understanding Cloud Foundry Security	450
Understanding Container Security	455
Understanding Container-to-Container Networking	459
Cloud Foundry Components	463
Component: Cloud Controller	466
Component: Messaging (NATS)	467
Component: Gorouter	469
Component: User Account and Authentication (UAA) Server	471
Component: Garden	480
HTTP Routing	481
Component: Droplet Execution Agent	484
Component: DEA Placement Algorithm	488
Component: Warden	492
Diego Architecture	499
Understanding Application SSH	504
How the Diego Auction Allocates Jobs	505
Operator's Guide	508

Understanding the Elastic Runtime Network Architecture	509
Load Balancer	509
Router	509
Identifying the API Endpoint for your Elastic Runtime Instance	510
Configuring SSL/TLS Termination at HAProxy	511
Configuring Proxy Settings for All Applications	513
Configuring Application Security Groups for Email Notifications	515
Configuring SSH Access for PCF	517
Identifying Elastic Runtime Jobs Using vCenter	519
Configuring System Logging in Elastic Runtime	521
Configuring UAA Password Policy	524
Configuring Authentication and Enterprise SSO for Elastic Runtime	526
Switching Application Domains	532
Scaling Elastic Runtime	535
Scaling Down Your MySQL Cluster	538
Using Docker Trusted Registries	540
Custom Branding Apps Manager	543
Monitoring App and Service Instance Usage	547
Monitoring Instance Usage with Apps Manager	551
Deploying Diego for Windows	553
Operating Diego for Windows	559
Monitoring a Pivotal Cloud Foundry Deployment	561
Providing a Certificate for Your SSL/TLS Termination Point	566
Enabling NFS Volume Services	569
Installing PCF Isolation Segment	570
Administering and Operating Cloud Foundry	574
Managing Custom Buildpacks	575
Adding a Custom Stack	577
Using Docker in Cloud Foundry	579
Managing Domains and Routes	581
Creating and Managing Users with the cf CLI	582
Creating and Managing Users with the UAA CLI (UAAC)	584
Creating and Modifying Quota Plans	590
Getting Started with the Notifications Service	594
Application Security Groups	596
Administering Container-to-Container Networking	603
Managing Isolation Segments	605
Feature Flags	610
Managing Diego Cell Limits During Upgrade	612
Setting a Maximum Number of Started Containers	614
Enabling IPv6 for Hosted Applications	615
Securing Traffic into Cloud Foundry	616
Enabling TCP Routing	621
Troubleshooting TCP Routes	624
Supporting WebSockets	627
Configuring Load Balancer Healthchecks for Cloud Foundry Routers	629
Troubleshooting Slow Requests in Cloud Foundry	631
Router Idle Keepalive Connections	634
Using Windows Cells (BETA)	635
Understanding Windows Cells	637

Understanding Stemcell Security	639
Building a Windows Stemcell	640
Deploying BOSH Release for Windows	643
Upgrading Windows Cells	645
Rotating Credentials in Garden Windows	647
Troubleshooting Windows Cells	648
Deploying .NET Apps to Windows Cells	653
Using Apps Manager	656
Getting Started with Apps Manager	657
Managing Orgs and Spaces Using Apps Manager	658
Managing User Roles with Apps Manager	661
Managing Apps and Service Instances Using Apps Manager	666
Viewing ASGs in Apps Manager	675
Configuring Spring Boot Actuator Endpoints for Apps Man	677
Using Spring Boot Actuators with Apps Manager	680
Cloud Foundry Command Line Interface (cf CLI)	684
Installing the cf CLI	685
Getting Started with the cf CLI	688
Using the cf CLI with an HTTP Proxy Server	693
Using the cf CLI with a Self-Signed Certificate	696
Using cf CLI Plugins	697
Developing cf CLI Plugins	699
Cloud Foundry CLI Reference Guide	700
Developer Guide	707
Considerations for Designing and Running an Application in the Cloud	709
Deploy an Application	711
Deploying a Large Application	715
Starting, Restarting, and Restaging Applications	717
Application Container Lifecycle	719
Routes and Domains	720
Changing Stacks	730
Deploying with Application Manifests	731
Using Application Health Checks	742
Scaling an Application Using cf scale	744
Running Tasks	745
Cloud Foundry Environment Variables	748
Using Blue-Green Deployment to Reduce Downtime and Risk	755
Troubleshooting Application Deployment and Health	758
Application SSH Overview	763
Accessing Apps with SSH	765
Accessing Services with SSH	770
Trusted System Certificates	772
Cloud Controller API Client Libraries	773
Delivering Service Credentials to an Application	774
Managing Service Instances with the cf CLI	776
Managing Service Keys	781
User-Provided Service Instances	783
Streaming Application Logs to Log Management Services	785
Service-Specific Instructions for Streaming Application Logs	787
Streaming Application Logs to Splunk	794

Streaming Application Logs with Fluentd	796
Configuring Play Framework Service Connections	797
Migrating a Database in Cloud Foundry	798
Using an External File System (Volume Services)	800
Security Guide	802
Security Concepts	804
PCF Security Processes	805
Pivotal Cloud Foundry Security Overview and Policy	806
PCF Testing, Release, and Security Lifecycle	808
Identity Management	811
PCF Infrastructure Security	812
Regenerating and Rotating Non-Configurable TLS/SSL Certificates	813
Stemcell Hardening FAQ	815
Network Security	819
TLS Connections in PCF Deployments	820
Security-Related PCF Tiles	821
Other Security Topics	822
Security Guidelines for Your IaaS Provider	823
How to Use This Topic	823
Buildpacks	824
Binary Buildpack	826
Go Buildpack	828
Java Buildpack	832
Getting Started Deploying Grails Apps	833
Getting Started Deploying Ratpack Apps	840
Getting Started Deploying Spring Apps	846
Tips for Java Developers	853
Configuring Service Connections for Grails	860
Configuring Service Connections for Play Framework	863
Configuring Service Connections for Spring	864
Cloud Foundry Eclipse Plugin	872
Cloud Foundry Java Client Library	891
BOSH Custom Trusted Certificate Support	894
Node.js Buildpack	895
Tips for Node.js Applications	898
Environment Variables Defined by the Node Buildpack	901
Configuring Service Connections for Node.js	902
.NET Core Buildpack	904
PHP Buildpack	908
Composer	911
Sessions	913
New Relic	914
PHP Buildpack Configuration	915
Deploying and Developing PHP Apps	918
Tips for PHP Developers	921
Python Buildpack	922
Ruby Buildpack	924
Getting Started Deploying Ruby Apps	927
Getting Started Deploying Ruby on Rails Apps	932
Deploy a Sample Ruby on Rails Application	935

Configure Rake Tasks for Deployed Apps	937
Tips for Ruby Developers	938
Environment Variables Defined by the Ruby Buildpack	944
Configure Service Connections for Ruby	945
Staticfile Buildpack	948
Using Buildpacks	952
Buildpack Detection	953
Proxy Usage	954
Supported Binary Dependencies	955
Configuring a Production Server	956
Developing Buildpacks	958
Custom Buildpacks	959
Packaging Dependencies for Offline Buildpacks	964
Merging from Upstream Buildpacks	967
Upgrading Dependency Versions	968
CF Buildpack Team CI	973
Releasing a New Buildpack Version	974
Updating Buildpack-related Gems	976
Services	977
Overview	978
Service Broker API v2.11	979
Table of Contents	979
Service Broker API Release Notes	995
Service Broker API Release Notes	995
Managing Service Brokers	997
Access Control	1001
Catalog Metadata	1004
Dashboard Single Sign-On	1008
Example Service Brokers	1012
Binding Credentials	1013
Application Log Streaming	1015
Route Services	1016
Supporting Multiple Cloud Foundry Instances	1021
Logging and Metrics	1022
Overview of the Loggregator System	1023
Using Loggregator	1023
Loggregator Components	1023
Loggregator Guide for Cloud Foundry Operators	1025
Application Logging in Cloud Foundry	1026
Security Event Logging for Cloud Controller and UAA	1030
Cloud Foundry Component Metrics	1034
Deploying a Nozzle to the Loggregator Firehose	1049
Cloud Foundry Data Sources	1054
Installing the Loggregator Firehose Plugin for cf CLI	1055
Pivotal Cloud Foundry Release Notes	1056
Pivotal Elastic Runtime v1.10 Release Notes	1057
Pivotal Cloud Foundry Ops Manager v1.10 Release Notes	1062
PCF Isolation Segment v1.10 Release Notes	1065
Stemcell Release Notes	1066

Installing Pivotal Cloud Foundry

 **Note:** Pivotal Cloud Foundry (PCF) for vCloud Air and vCloud Director is deprecated and availability is restricted to existing customers. Contact [Support](#) for more information.

 **Note:** PCF 1.10 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 by following the procedure in [Regenerating and Rotating Non-Configurable TLS/SSL Certificates](#).

Welcome to [Pivotal Cloud Foundry](#)!

The following IaaS-specific guides are intended to walk you through the process of getting your Pivotal Cloud Foundry (PCF) deployment up and running.

 Check out the 15-minute [Getting Started with PCF](#) tutorial for learning Pivotal Cloud Foundry app deployment concepts.

If you experience a problem while following the steps below, check the [Known Issues](#), or refer to the [PCF Troubleshooting Guide](#).

Once you have completed the steps in this guide, explore the documentation on [docs.pivotal.io](#) to learn more about [Pivotal Cloud Foundry](#) and the Pivotal product suite.

Prepare for Installation:

- [Preparing Your Firewall for Deploying Pivotal Cloud Foundry](#)
- [Pivotal Cloud Foundry IaaS User Role Guidelines](#)

Install Pivotal Cloud Foundry:

- [Installing Pivotal Cloud Foundry on AWS](#)
- [Installing Pivotal Cloud Foundry on Azure](#)
- [Installing Pivotal Cloud Foundry on GCP](#)
- [Installing Pivotal Cloud Foundry on OpenStack](#)
- [Installing Pivotal Cloud Foundry on vSphere](#)

Preparing Your Firewall for Deploying Pivotal Cloud Foundry

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 Elastic Runtime 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
- **22 (Optional)**: Only necessary if you want to connect using SSH

For more information about required ports for additional installed products, refer to the 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 PCF_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
```

- 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 Elastic Runtime from accessing `*.xip.io`.

To verify that Elastic Runtime can correctly resolve DNS entries:

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

Pivotal Cloud Foundry IaaS User Role 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 Elastic Runtime 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 [Guidelines for Creating User Roles for PCF on AWS](#) 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 Elastic Runtime.

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.

Installing Pivotal Cloud Foundry on AWS

Page last updated:

This topic describes how to install [Pivotal Cloud Foundry](#) (PCF) on Amazon Web Services (AWS) using the PCF CloudFormation template.

 **Note:** If you are performing an upgrade to PCF 1.10, see [Upgrading Pivotal Cloud Foundry](#) for critical upgrade information.

The CloudFormation template for PCF describes the set of necessary AWS resources and properties. When you create an AWS stack using the PCF template, CloudFormation provisions all the infrastructure that you need to deploy PCF on AWS.

Pivotal strongly recommends using CloudFormation to install PCF on AWS. If you cannot use CloudFormation for your installation, contact [Pivotal Support](#).

 **Note:** The CloudFormation template for Elastic Runtime includes a reference to another CloudFormation template for Ops Manager. For more information about how IaaS user roles are configured for each template, see the [Pivotal Cloud Foundry IaaS User Role Guidelines](#) topic.

Prerequisites

The following sections describe general requirements for running PCF and specific requirements for running PCF on AWS.

General Requirements

The following are general requirements for deploying and managing a PCF deployment with Ops Manager and Elastic Runtime:

- **(Recommended)** Ability to create a wildcard DNS record to point to your router or load balancer. Alternatively, you can use a service such as xip.io. For example, `203.0.113.0.xip.io`.
Elastic Runtime 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 router, every application deployed to the `example.com` domain resolves to the IP address of your router.
- **(Recommended)** A network without DHCP available for deploying the Elastic Runtime VMs

 **Note:** If you have DHCP, refer to the Troubleshooting Guide to avoid issues with your installation.

- Sufficient IP allocation:
 - One IP address for each VM instance
 - An additional IP address for each instance that requires static IPs
 - An additional IP address for each errand
 - An additional IP address for each compilation worker: `IPs needed = VM instances + static IPs + errands + compilation workers`

 **Note:** BOSH requires that you allocate a sufficient number of additional dynamic IP addresses when configuring a reserved IP range during installation. BOSH uses these IPs during installation to compile and deploy VMs, install Elastic Runtime, and connect to services. We recommend that you allocate at least 36 dynamic IP addresses when deploying Ops Manager and Elastic Runtime.

- The most recent version of the [Cloud Foundry Command Line Interface \(cf CLI\)](#)
- One or more NTP servers if not already provided by your IaaS

AWS Requirements

The following are the minimum resource requirements for maintaining a [Pivotal Cloud Foundry](#) (PCF) deployment with Ops Manager and Elastic Runtime on Amazon Web Services infrastructure:

- 1 Elastic Load Balancer

- 1 Relational Database Service. We recommend at least a db.m3.xlarge instance with 100 GB of allocated storage.
- 5 S3 Buckets
- EC2 Instances:
 - 13 t2.micro
 - 15 t2.small
 - 2 m3.medium
 - 6 m3.xlarge
 - 3 m3.2xlarge

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](#).

Install PCF using CloudFormation

Complete the following procedures to install PCF using CloudFormation:

1. [Deploying the CloudFormation Template for PCF on AWS](#)
2. [Launching an Ops Manager Director Instance on AWS](#)
3. [Configuring Ops Manager Director on AWS](#)
4. [\(Optional\) Installing the PCF IPsec Add-On](#)
5. [Deploying Elastic Runtime on AWS](#)

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](#)

Additional AWS Configuration

See the following topics for additional AWS configuration information:

- [Guidelines for Creating User Roles on AWS](#)
- [Configuring Amazon EBS Encryption](#)
- [Creating a Proxy ELB for Diego SSH without CloudFormation](#)

Deploying the CloudFormation Template for Pivotal Cloud Foundry on AWS

Page last updated:

This topic describes how to deploy the CloudFormation template for [Pivotal Cloud Foundry](#) (PCF) on Amazon Web Services (AWS).

An AWS CloudFormation template describes a set of AWS resources and properties. Follow the instructions below to use a CloudFormation template to create the infrastructure that you need to deploy PCF to AWS.

The template is designed to output the resources necessary for two availability zones (AZ), with a private and public subnet designated for each AZ. The Elastic Load Balancer will be attached to the public subnet of both AZs to balance traffic across both environments. Three AZs is actually recommended as the desired number of AZs for a highly available deployment of PCF, however many AWS regions only have two AZs available.

Note: The CloudFormation template for Elastic Runtime includes a reference to another CloudFormation template for Ops Manager. For more information about how IaaS user roles are configured for each template, refer to the [Pivotal Cloud Foundry IaaS User Role Guidelines](#) topic.

Note: Before following the procedure below, confirm that you have selected the correct region within your AWS account. All of the AWS resources for your deployment must exist within a single region. See the [Amazon documentation on regions and availability zones](#) for help selecting the correct region for your deployment.

Step 1: Download the PCF CloudFormation Template

1. Sign in to [Pivotal Network](#).
2. Select **Elastic Runtime**. From the **Releases** drop-down menu, select the release that you wish to install.
3. Download the **PCF CloudFormation for AWS Setup**
4. Save the file as `pcf.json`.

Step 2: Upload an SSL Certificate to AWS

You can add an SSL Certificate using two methods:

- The [AWS CLI](#)
- The [AWS Certificate Manager](#)

(Option) Create SSL Certificate using the AWS CLI

The [AWS CLI](#) must be installed on your machine and configured to a user account with admin access privileges on your AWS account.

1. Obtain or create an SSL server certificate. For more information, see the [AWS documentation on SSL certificates](#). When you create a certificate signing request (CSR) in the “Create a Server Certificate” instructions, you must use your wildcard domain as the Common Name input.
2. Add the following additional domains and wildcards using the OpenSSL SAN (subjectAltName) extension:

- `*.system.example.com`
- `*.login.system.example.com`
- `*.uaa.system.example.com`
- `*.apps.example.com`

Note: If you use a self-signed certificate or select the “Generate Self-Signed RSA Certificate” option during the [Deploying Elastic Runtime on AWS](#) installation process, you can ignore the step above. However, make sure you upload the self-signed certificate to AWS and attach the certificate to the listeners on the AWS Elastic Load Balancer. Pivotal recommends only using

a self-signed certificate for testing and development.

- Upload your SSL certificate to AWS. For more information, see the [AWS documentation on uploading SSL certificate using the CLI](#).

```
$ aws iam upload-server-certificate \
--server-certificate-name YOUR-CERTIFICATE \
--certificate-body file://YOUR-PUBLIC-KEY-CERT-FILE.pem \
--private-key file://YOUR-PRIVATE-KEY-FILE.pem \
```

For example:

```
$ aws iam upload-server-certificate \
--server-certificate-name myServerCertificate \
--certificate-body file://my-certificate.pem \
--private-key file://my-private-key.pem \
```

 **Note:** If you receive an upload error (MalformedCertificate), run the following command to convert your server certificate to the PEM format as required by the AWS Identity and Management (IAM) service: `$ openssl x509 -inform PEM -in my-certificate.pem`. Then try your upload again.

- After successfully uploading the certificate to your AWS account, you will see output metadata for your certificate. For example:

```
{
  "ServerCertificateMetadata": {
    "ServerCertificateId": "ASCAI3HRFYUTD55KNAF64",
    "ServerCertificateName": "myServerCertificate",
    "Expiration": "2016-10-18T18:41:59Z",
    "Path": "/",
    "Arn": "arn:aws:iam::9240874958318:server-certificate/myServerCertificate",
    "UploadDate": "2015-10-19T19:10:57.404Z"
  }
}
```

- Record the value of the SSL Certificate **ARN** (Amazon Resource Name) to use when [configuring](#) your AWS resource stack. Alternatively, if you know the name of the certificate, you can run the following command to retrieve certificate metadata later:

```
$ aws iam get-server-certificate --server-certificate-name YOUR-CERT-NAME
```

For example:

```
$ aws iam get-server-certificate --server-certificate-name myServerCertificate
```

(Option) Create SSL Certificate using the AWS Certificate Manager

- Log into your AWS management console and navigate to **Certificate Manager**. If your Certificate Manager has no certificates, click **Get Started**.
- Under **Add domain names**, enter the following wildcard subdomains to the certificate, based on your domain (example: example.com). Click **Add another name to this certificate** until you have entered them all:
 - *.example.com
 - *.system.example.com
 - *.login.system.example.com
 - *.uaa.system.example.com
 - *.apps.example.com
- Click **Review and Request** to review, and **Confirm and Request** to confirm.
- Check the email account registered for the domain owner. Open the certificate approval email message sent from Amazon Certificates, and click the email link to the approval page for the SSL certificate.
- From the approval page, click **I Approve**.
- Record the SSL Certificate Amazon Resource Name (ARN) shown on the confirmation page to use when [configuring](#) your AWS resource stack. Alternatively, you can retrieve certificate metadata later by selecting the certificate listing in Certificate Manager and recording values in the **Details** pane that appears underneath.

Step 3: Create a Resource Stack Using the CloudFormation Template

1. Log in to the [AWS Console](#).
2. In the second column, under **Management Tools**, click **CloudFormation**.

The screenshot shows the AWS Management Console with the 'Services' menu open. Under the 'Management Tools' section, the 'CloudFormation' icon is circled in red.

- Compute**
 - EC2** Virtual Servers in the Cloud
 - EC2 Container Service** Run and Manage Docker Containers
 - Elastic Beanstalk** Run and Manage Web Apps
 - Lambda** Run Code in Response to Events
- Storage & Content Delivery**
 - S3** Scalable Storage in the Cloud
 - CloudFront** Global Content Delivery Network
 - Elastic File System** PREVIEW Fully Managed File System for EC2
 - Glacier** Archive Storage in the Cloud
 - Import/Export Snowball** Large Scale Data Transport
 - Storage Gateway** Integrates On-Premises IT Environments with Cloud Storage
- Database**
 - RDS** Managed Relational Database Service
- Developer Tools**
 - CodeCommit** Store Code in Private Git Repositories
 - CodeDeploy** Automate Code Deployments
 - CodePipeline** Release Software using Continuous Delivery
- Management Tools**
 - CloudWatch** Monitor Resources and Applications
 - CloudFormation** Create and Manage Resources with Templates (circled in red)
 - CloudTrail** Track User Activity and API Usage
 - Config** Track Resource Inventory and Changes
 - OpsWorks** Automate Operations with Chef
 - Service Catalog** Create and Use Standardized Products
 - Trusted Advisor** Optimize Performance and Security
- Security & Identity**
 - Identity & Access Management** Manage User Access and Encryption Keys

3. Click **Create New Stack**.

The screenshot shows the 'Create a Stack' wizard. The 'Upload a template to Amazon S3' option is selected, and the file 'pcf.json' is chosen.

Create a Stack

AWS Cloudformation allows you to quickly and easily deploy your infrastructure resources and applications on AWS. You can use one of the templates we provide to get started quickly with applications like WordPress or Drupal, one of the many sample templates or create your own template.

You do not currently have any stacks. Click the "Create New Stack" button below to create a new AWS Cloudformation Stack.

Create New Stack

4. Select **Upload a template to Amazon S3**.

The screenshot shows the 'Choose a template' step of the CloudFormation wizard. The 'Upload a template to Amazon S3' option is selected, and the file 'pcf.json' is chosen.

Choose a template A template is a JSON-formatted text file that describes your stack's resources and their properties. [Learn more.](#)

Select a sample template

Upload a template to Amazon S3 **Choose File** pcf.json

Specify an Amazon S3 template URL

5. Click **Browse**. Browse to and select the `pcf.json`, the **Pivotal Cloud Foundry CloudFormation script for AWS** file that you downloaded. Click **Next**.
6. On the next screen, name the stack `pcf-stack`.

7. In the **Specify Parameters** page, complete the following fields:

Parameters

01NATKeyPair	<input type="button" value="Search"/>	Select the SSH keypair to use for the NAT vm
02NATInstanceType	<input type="text" value="t2.medium"/>	Select the Instance Type to use for the NAT vm
03OpsManagerIngress	<input type="text" value="0.0.0.0/0"/>	CIDR range allowed to connect to Ops Manager instance
04RdsDBName	<input type="text" value="bosh"/>	BOSH database name
05RdsUsername	<input type="text"/>	BOSH database username
06RdsPassword	<input type="text"/>	BOSH database password
07SSLCertificateARN	<input type="text"/>	ARN for pre-uploaded SSL certificate
08OpsManagerTemplate	<input type="text" value="https://s3.amazonaws.com/ops-man-referenc"/>	S3 Location for OpsManager CloudFormation Template
09ElbPrefix	<input type="text"/>	Prefix for the name of the ELBs generated. NOTE: Leave empty to use default prefix of AWS::StackName
10AllowHttpOnElb	<input type="text" value="true"/>	Allow HTTP traffic on PCF-ELB port 80. Default: true.

Cancel Previous Next

- **01NATKeyPair:** Use the drop-down menu to select the name of your pre-existing AWS key pair. If you do not have a pre-existing key pair, create one in [AWS](#) and return to this step.
- **02NATInstanceType:** Do not change this value.
- **03OpsManagerIngress:** Do not change this value.

 **Note:** The first parameter name begins with `03`.

- **04RdsDBName:** Do not change this value.
- **05RdsUserName:** Enter a username for the RDS database.

 **Note:** Do not enter the username `rdsadmin`. AWS reserves the `rdsadmin` user account for internal database instance management.

- **06RdsPassword:** Enter a password for the RDS database.
- **07SSLCertificateARN:** Enter your uploaded SSL Certificate ARN.
- **08OpsManagerTemplate:** The default template link provided here works. Otherwise you can enter your own S3 bucket location of the Ops Manager CloudFormation script.
- **09ElbPrefix:** Prefix for the generated names of the ELBs. Any string you specify in this field will be prefixed to `-pcf-elb` to form the name of your ELBs. Leave empty to use the default prefix of `AWS::StackName`.
- **10AllowHttpOnElb:** Set this to `true` to listen for HTTP traffic on port `80`. This is the default. Set it to `false` to only listen for traffic on ports `443` and `4443`.

8. Click **Next**.

9. On the **Options** page, leave the fields blank and click **Next**.

Options

Tags

You can specify tags (key-value pairs) for resources in your stack.
You can add up to 10 unique key-value pairs for each stack.

	Key (127 characters maximum)	Value (255 characters maximum)	
1	<input type="text"/>	<input type="text"/>	+

Advanced

You can set additional options for your stack, like notification options and a stack policy.

[Cancel](#) [Previous](#) [Next](#)

- On the **Review** page, select the **I acknowledge that this template might cause AWS CloudFormation to create IAM resources** checkbox and click **Create**.

i The following resource(s) require capabilities: [AWS::IAM::User, AWS::IAM::Policy, AWS::IAM::AccessKey]

This template might include Identity and Access Management (IAM) resources, which can include groups, IAM users, and IAM roles with certain permissions. Ensure that the template you are using is from a trusted source. [Learn more](#).

I acknowledge that this template might cause AWS CloudFormation to create IAM resources.

[Cancel](#) [Previous](#) [Create](#)

AWS runs the CloudFormation script and creates the infrastructure that you need to deploy PCF to AWS. This may take a few moments. You can click on the **Events** tab to view the progress of the setup.

When the installation process successfully completes, AWS displays **CREATE_COMPLETE** as the status of the stack.

Create Stack Actions ▾ Design template C ⚙

Filter: Active ▾ By Name: Showing 2 stacks

Stack Name	Created Time	Status	Description
<input type="checkbox"/> pcf-stack-OpsManStack-1FUM	2016-07-12 12:09:45 UTC-0700	CREATE_COMPLETE	Cloudformation template for configuring Pivotal Cloud Foundry on AWS
<input checked="" type="checkbox"/> pcf-stack	2016-07-12 12:09:40 UTC-0700	CREATE_COMPLETE	Cloudformation template for configuring Pivotal Cloud Foundry on AWS

Overview Outputs Resources Events Template Parameters Tags Stack Policy Change Sets

Key	Value	Description
PcfPublicSubnetAvailabilityZone	us-west-1b	
PcfPrivateSubnetAvailabilityZone	us-west-1b	
PcfPrivateSubnet2AvailabilityZone	us-west-1c	
PcfElasticRuntimeS3DropletsBucket	pcf-stack-felasticruntimes3dropletsbucket-1b9ij48t4v29	
PcfElbSshDnsName	pcf-stack-f-ssh-elb-1859745201.us-west-1.elb.amazonaws.com	
PcfVmsSecurityGroupId	sg-043a5c60	
PcfElasticRuntimeS3BuildpacksBucket	pcf-stack-felasticruntimes3buildpacksbucket-kbx64zosvzs	
PcfVpc	vpc-36715a53	
PcfPublicSubnetId	subnet-100e3e49	
PcfRdsPassword	bosh1234	
PcfIamUserName	pcf-stack-psManStack-1AGSRAPW2HVO5-PcfiamUser-IC0JC0UZOW00	
PcfPrivateSubnet2Id	subnet-a79e74c3	
PcfRdsDBName	bosh	
PcfOpsManagerS3Bucket	pcf-stack-osmanstack-1agsrap-pcfopsmanagers3bucket-ky	
PcfPublicSubnetId2	subnet-a09e74c4	
PcfElasticRuntimeS3PackagesBucket	pcf-stack-felasticruntime3packagess3bucket-ucosfthbun	

After completing this procedure, complete all of the steps in the following topics:

- [Launching an Ops Manager Director Instance on AWS](#)
- [Configuring Ops Manager Director for AWS](#)
- [Deploying Elastic Runtime on AWS](#)

Return to [Installing Pivotal Cloud Foundry Using AWS CloudFormation](#).

Launching an Ops Manager Director Instance on AWS

Page last updated:

This topic describes how to deploy Ops Manager Director after deploying the CloudFormation template for [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 [Deploying the CloudFormation Template for PCF on AWS](#) topic. After you complete this procedure, follow the instructions in the [Configuring Ops Manager Director on AWS CloudFormation](#) and [Configuring Elastic Runtime on AWS CloudFormation](#) topics.

Step 1: Open the Outputs Tab in AWS Stacks

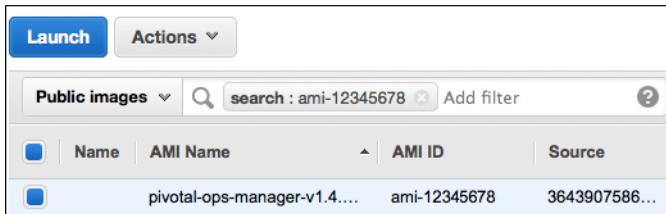
1. In the dashboard of your [AWS Console](#), click **CloudFormation**. The Stacks Dashboard displays.
2. Select the **pcf-stack** checkbox, then select the **Outputs** tab.

Key	Value	Description
PcfPublicSubnetAvailabilityZone	us-west-1b	
PcfPrivateSubnetAvailabilityZone	us-west-1b	
PcfPrivateSubnet2AvailabilityZone	us-west-1c	
PcfElasticRuntimeS3DropletsBucket	pcf-stack-felasticruntimes3dropletsbucket-1lb9ij48t4v29	
PcfElbSshDnsName	pcf-stack-f-ssh-elb-1859745201.us-west-1.elb.amazonaws.com	
PcfVmsSecurityGroupId	sg-043a5c60	
PcfElasticRuntimeS3BuildpacksBucket	pcf-stack-felasticruntimes3buildpacksbucket-kbx64zosvzs	
PcfVpc	vpc-36715a53	
PcfPublicSubnetId	subnet-100e3e49	
PcfRdsPassword	bosh1234	
PcfIamUserName	pcf-stack-psManStack-1AGSRAPW2HVO5-PcfIamUser-IC0JCQUZOW00	
PcfPrivateSubnet2Id	subnet-a79e74c3	
PcfRdsDBName	bosh	
PcfOpsManagerS3Bucket	pcf-stack-osmanstack-1agsrap-pcfopsmanagers3bucket-kypy81lfqlas	
PcfPublicSubnetId2	subnet-a09e74c4	
PcfElasticRuntimeS3BuildpacksBucket2	pcf-stack-felasticruntimes3buildpacksbucket-1nccfthiuu	

In the steps described below, use the information from the **Value** column of the **Outputs** tab to configure your PCF installation.

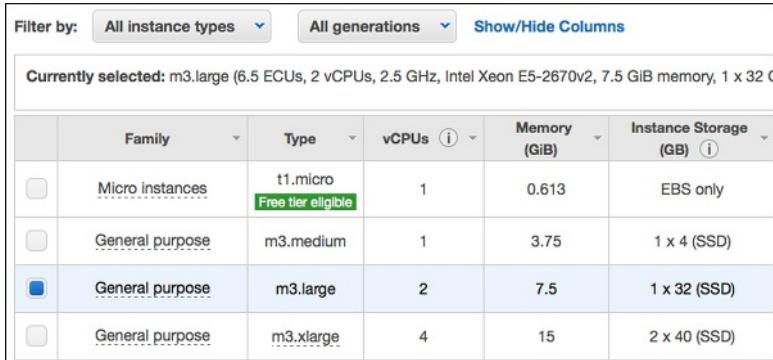
Step 2: Select a Pivotal Ops Manager AMI Instance

1. Log in to the [Pivotal Network](#) and click Pivotal Cloud Foundry **Ops Manager**.
2. From the **Releases** dropdown, select the release to install.
3. Select **Pivotal Cloud Foundry Ops Manager for AWS** to download the [OpsManagerx.x.xonAWSFulfillmentInstructions.pdf](#) file. This document lists AMI IDs for Pivotal Ops Manager for specific regions.
4. Log in to the [AWS Console](#). Navigate to the EC2 Dashboard.
5. In the left navigation panel, click **AMIs**.
6. Using the [OpsManagerx.x.xonAWSFulfillmentInstructions.pdf](#) document, enter the AMI ID for your AWS region in the Public images search field. This search locates the appropriate Pivotal Ops Manager AMI for your region within public images.



A screenshot of the AWS EC2 AMI search interface. At the top, there are two buttons: "Launch" and "Actions". Below that is a search bar with the placeholder "search : ami-12345678" and a "Add filter" button. The main area shows a table with columns: "Name", "AMI Name", "AMI ID", and "Source". There is one row visible with the following data:

Name	AMI Name	AMI ID	Source
<input checked="" type="checkbox"/>	pivotal-ops-manager-v1.4....	ami-12345678	3643907586...

7. Select this AMI and click **Launch**.
 8. Choose **m3.large** for your instance type.
- 
- A screenshot of the AWS EC2 instance type selection interface. At the top, there are filters: "Filter by: All instance types" and "All generations". Below that is a table header with columns: "Family", "Type", "vCPUs", "Memory (GiB)", and "Instance Storage (GB)". The table data is as follows:
- | Family | Type | vCPUs | Memory (GiB) | Instance Storage (GB) |
|---|---|-------|--------------|-----------------------|
| Micro instances | t1.micro
<small>Free tier eligible</small> | 1 | 0.613 | EBS only |
| General purpose | m3.medium | 1 | 3.75 | 1 x 4 (SSD) |
| <input checked="" type="checkbox"/> General purpose | m3.large | 2 | 7.5 | 1 x 32 (SSD) |
| General purpose | m3.xlarge | 4 | 15 | 2 x 40 (SSD) |

9. Click **Next: Configure Instance Details**.

Step 3: Configure Instance Details

1. Complete the **Config Instance Details** page with information from the [Outputs tab](#) in the AWS Stacks Dashboard:

Number of instances (1) 1

Purchasing option (i) Request Spot Instances

Network (i) vpc-XXXXXXXXXX (10.0.0.0/16) | pcf-vpc C Create new VPC

Subnet (i) subnet-XXXXXXXXXX (10.0.0.0/24) | us-east-1a C Create new subnet
249 IP Addresses available

Auto-assign Public IP (i) Enable

IAM role (i) None C Create new IAM role

Shutdown behavior (i) Stop

Enable termination protection (i) Protect against accidental termination

Monitoring (i) Enable CloudWatch detailed monitoring
Additional charges apply.

EBS-optimized instance (i) Launch as EBS-optimized instance
Additional charges apply.

Tenancy (i) Shared tenancy (multi-tenant hardware)
Additional charges will apply for dedicated tenancy.

Network interfaces

Device	Network Interface	Subnet	Primary IP	Secondary IP addresses
eth0	New network interf	subnet- XXXXXXXXXX	Auto-assign	Add IP

Advanced Details

Review and Launch

- Select the **Network** that matches the value of **PcfVpc**.
 - Select the **Subnet** that matches the value of **PcfPublicSubnetId**.
2. Set **Auto-assign Public IP** to **Enable**.
3. Click **Next: Add Storage**.
4. On the **Add Storage** page, adjust the **Size (GiB)** value. Pivotal recommends increasing this value to a minimum of 100 GB.

Type (i)	Device (i)	Snapshot (i)	Size (GiB) (i)	Volume Type (i)	IOPS (i)
Root	/dev/sda1	snap- XXXXXXXXXX	100	General Purpose (SSD)	300 / 3000
Instance Store 0 :	/dev/sdb	N/A	N/A	N/A	N/A

Add New Volume

Review and Launch

5. Click **Next: Tag Instance**.
6. On the **Tag Instance** page, add a **Key** Name with **Value** Ops Manager.

Key (127 characters maximum)	Value (255 characters maximum)
Name	Ops Manager

Create Tag (Up to 10 tags maximum)

Review and Launch

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

Step 4: Configure Security Group

- Select the **Select an existing security group** option.
- Select the **Security Group ID** that matches the value of **PcfOpsManagerSecurityGroupId** located in the **Outputs** tab of the Stacks

Assign a security group: Create a new security group Select an existing security group

Security Group ID	Name	Description	Actions
sg-26716d42	default	default VPC security group	Copy to new
sg-3a716d5e	pcf-stack-PcfMysqlSecurityGroup-1K7ZNDGY2FU0H	PCF MySQL Security Group	Copy to new
sg-39716d5d	pcf-stack-PcfNatSecurityGroup-10T80HF3NLTD0	NAT Security Group	Copy to new
sg-04716d60	pcf-stack-PcfOpsManagerSecurityGroup-OEGO	Ops Manager Security Group	Copy to new
sg-06716d62	pcf-stack-PcfVmsSecurityGroup-GYPF6VACFYMD	PCF VMs Security Group	Copy to new

Inbound rules for sg-04716d60 (Selected security groups: sg-04716d60)

Type	Protocol	Port Range	Source
SSH	TCP	22	0.0.0.0/0
SSH	TCP	22	10.0.0.0/16
HTTP	TCP	80	0.0.0.0/0
Custom TCP Rule	TCP	6868	10.0.0.0/16
Custom TCP Rule	TCP	25555	10.0.0.0/16
HTTPS	TCP	443	0.0.0.0/0

[Cancel](#) [Previous](#) **Review and Launch**

dashboard.

3. Click **Review and Launch**.

Step 5: Deploy Ops Manager

1. Review the instance launch details. Click **Launch**.

Step 7: Review Instance Launch
Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

AMI Details [Edit AMI](#)

pivotal-ops-manager-v1.4.2.0-RC1 - ami-a03126c8
PCF Ops Manager for AWS - 1.4.2.0
Root Device Type: ebs Virtualization type: paravirtual

Instance Type [Edit instance type](#)

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
m3.large	6.5	2	7.5	1 x 32	-	Moderate

Security Groups [Edit security groups](#)

Security Group ID	Name	Description
sg-04716d60	pcf-stack-PcfOpsManagerSecurityGroup-OEGO	Ops Manager Security Group

All selected security groups inbound rules

Security Group ID	Type	Protocol	Port Range	Source
sg-04716d60	SSH	TCP	22	0.0.0.0/0
sg-04716d60	SSH	TCP	22	10.0.0.0/16
sg-04716d60	HTTP	TCP	80	0.0.0.0/0
sg-04716d60	Custom TCP Rule	TCP	6868	10.0.0.0/16
sg-04716d60	Custom TCP Rule	TCP	25555	10.0.0.0/16
sg-04716d60	HTTPS	TCP	443	0.0.0.0/0

Instance Details [Edit instance details](#)

Storage [Edit storage](#)

Tags [Edit tags](#)

[Cancel](#) [Previous](#) **Launch**

2. Use the first drop-down menu to select **Choose an existing key pair**. Use the second drop-down menu to select the name of your pre-existing AWS key pair.
3. Select the acknowledgement checkbox.
4. Click **Launch Instances**. If successful, you will see the **Launch Status Page**.

Select an existing key pair or create a new key pair X

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

Choose an existing key pair pcf

Select a key pair pcf

I acknowledge that I have access to the selected private key file (pcf.pem), and that without this file, I won't be able to log into my instance.

Launch Instances Cancel

5. Click **View Instances**. Or alternately, navigate to **Instances** from the left navigation panel of the EC2 Dashboard.
6. AWS deploys Ops Manager. This may take a few minutes. When complete, AWS displays an **Instance State** of running and a **Status Check** of passed when the Ops Manager deployment successfully completes.

Instance ID	Instance Type	Instance State	Status Checks	Public DNS	Public IP	Key Name
i-0003b6f	m3.xlarge	running	2/2 checks passed	ec2-62-173-3-213.amazonaws.com	52.173.3.213	pcf
i-0003b6g	m3.xlarge	running	2/2 checks passed	ec2-62-173-3-214.amazonaws.com	52.173.3.214	pcf

Step 6: Create a DNS Entry

Note: For security, Ops Manager 1.7 and later require that you log in using a fully qualified domain name during the [initial configuration](#).

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 Configure Ops Manager Director for AWS step below.

Step 7: Configure Ops Manager Director for AWS

After you complete this procedure, follow the instructions in the [Configuring Ops Manager Director on AWS CloudFormation](#) and [Configuring Elastic Runtime on AWS CloudFormation](#) topics.

Return to [Installing Pivotal Cloud Foundry Using AWS CloudFormation](#).

Configuring Ops Manager Director on AWS

Page last updated:

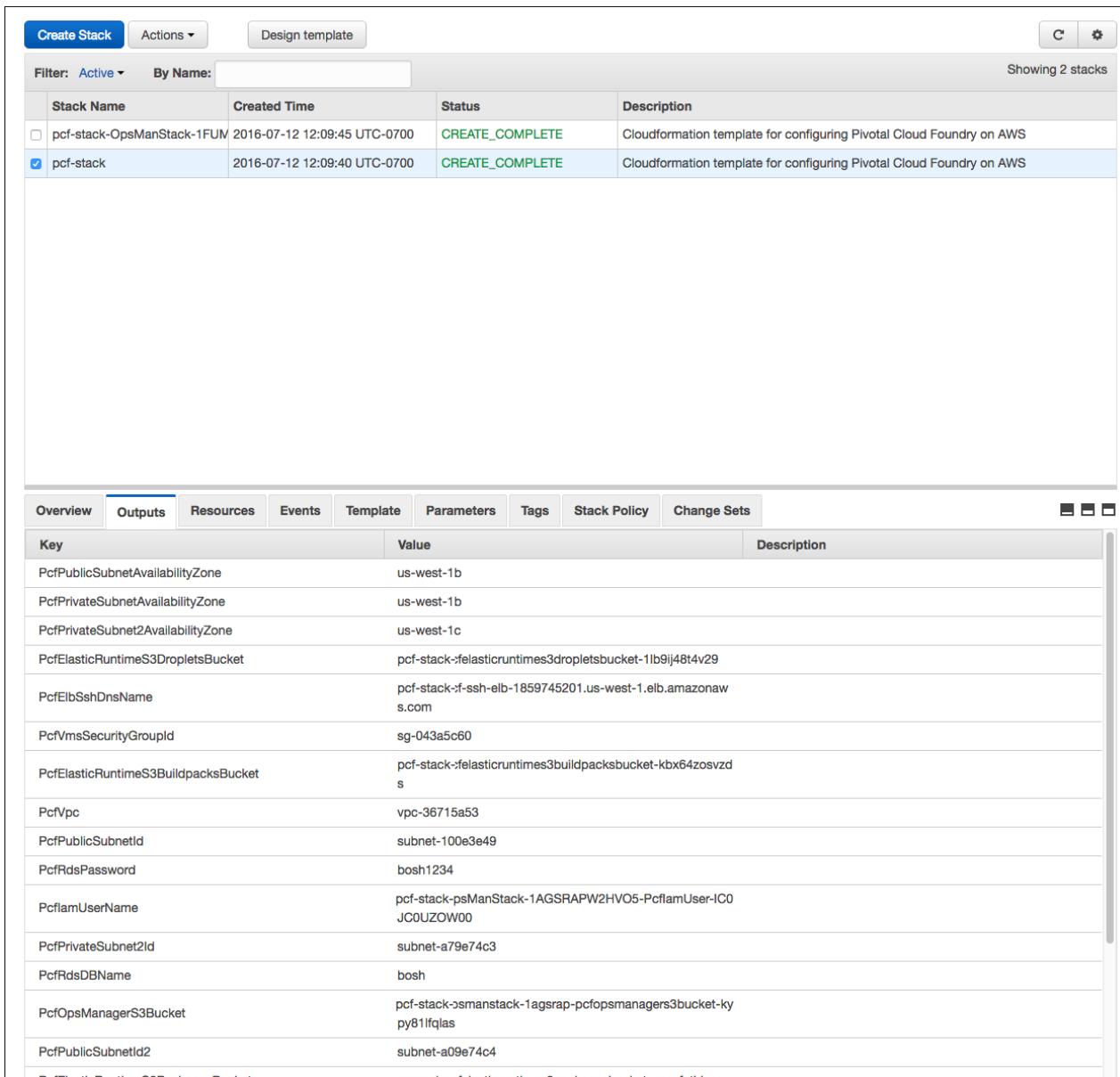
This topic describes how to configure the Ops Manager Director after deploying the CloudFormation template for Pivotal Cloud Foundry (PCF) on Amazon Web Services (AWS). Use this topic when [Installing Pivotal Cloud Foundry Using AWS CloudFormation](#).

Before beginning this procedure, ensure that you have successfully completed all steps in the [Deploying the CloudFormation Template for PCF on AWS](#) and the [Launching an Ops Manager Director Instance on AWS CloudFormation](#) topics. After you complete this procedure, follow the instructions in the [Deploying Elastic Runtime on AWS CloudFormation](#) 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: Open the Outputs Tab in AWS Stacks

1. In the dashboard of your [AWS Console](#), click **CloudFormation**. The Stacks Dashboard displays.
2. Select the **pcf-stack** checkbox, then select the **Outputs** tab.



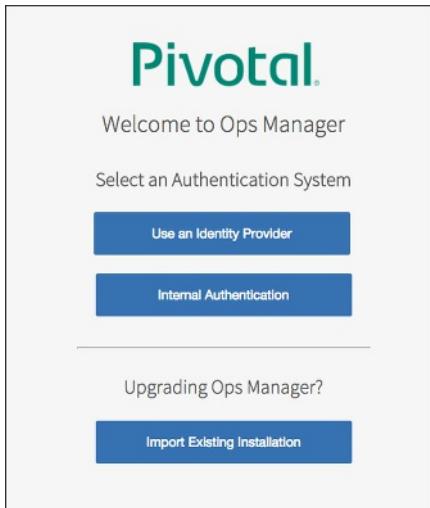
Key	Value	Description
PcfPublicSubnetAvailabilityZone	us-west-1b	
PcfPrivateSubnetAvailabilityZone	us-west-1b	
PcfPrivateSubnet2AvailabilityZone	us-west-1c	
PcfElasticRuntimeS3DropletsBucket	pcf-stack-felasticruntimes3dropletsbucket-1lb9ij48t4v29	
PcfElbSshDnsName	pcf-stack-f-ssh-elb-1859745201.us-west-1.elb.amazonaws.com	
PcfVmsSecurityGroupId	sg-043a5c60	
PcfElasticRuntimeS3BuildpacksBucket	pcf-stack-felasticruntimes3buildpacksbucket-kbx64zosvzs	
PcfVpc	vpc-36715a53	
PcfPublicSubnetId	subnet-100e3e49	
PcfRdsPassword	bosh1234	
PcfFlamUserName	pcf-stack-psManStack-1AGSRAPW2HVO5-PcfFlamUser-IC0JC0UZOW00	
PcfPrivateSubnet2Id	subnet-a79e74c3	
PcfRdsDBName	bosh	
PcfOpsManagerS3Bucket	pcf-stack-opsmanstack-1agsrap-pcfopsmanagers3bucket-kypy81fqlas	
PcfPublicSubnetId2	subnet-a09e74c4	
PcfElasticRuntimeS3BuildpacksBucket	pcf-stack-felasticruntimes3buildpacksbucket-1aefaf1bc	

In the steps described below, use the information from the **Value** column of the **Outputs** tab to configure your PCF installation.

Step 2: Access Ops Manager

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

1. In a web browser, navigate to the fully qualified domain you created in the [Create a DNS Entry](#) step of [Launching an Ops Manager Director Instance on AWS](#).
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 are using 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.

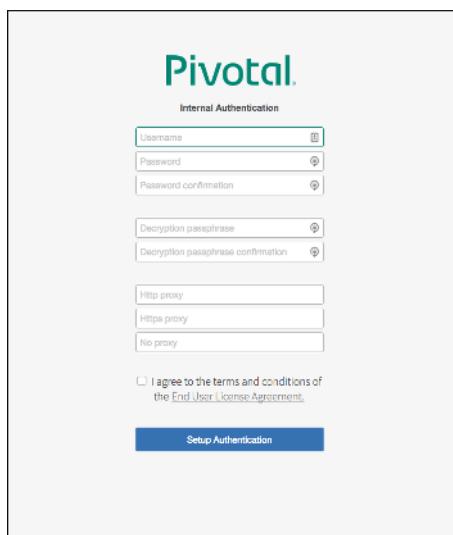
- **Single sign on URL:** <https://BOSH-IP:8443/saml/SSO/alias/BOSH-IP>
- **Audience URI (SP Entity ID):** <https://BOSH-IP:8443>
- **Name ID:** Email Address
- SAML authentication requests are always signed

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

Internal Authentication

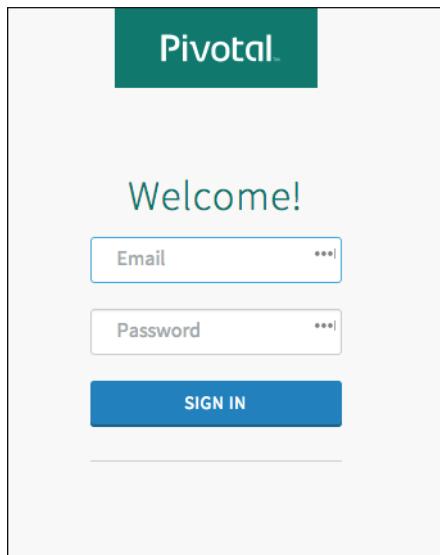
1. When redirected to the **Internal Authentication** page, you must complete the following steps:

- 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**.



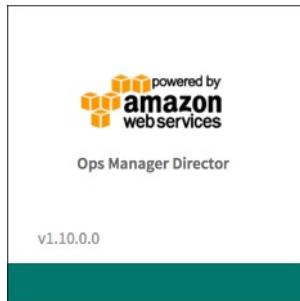
The screenshot shows the 'Internal Authentication' setup page. It includes fields for 'Username', 'Password', 'Password confirmation', 'Decryption passphrase', 'Decryption passphrase confirmation', and proxy selection ('Http proxy', 'Https proxy', 'No proxy'). A checkbox for accepting the 'End User License Agreement' is present, along with a 'Setup Authentication' button at the bottom.

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



Step 3: AWS Config Page

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



2. Select **AWS Config** to open the **AWS Management Console Config** page.

[Installation Dashboard](#)

Ops Manager Director

Settings Status Credentials

AWS Config

AWS Management Console Config

Use AWS Keys

Access Key ID*

AWS Secret Key*

Use AWS Instance Profile

AWS IAM Instance Profile*

VPC ID*

Security Group ID* The ID of the security group that will be assigned to your Ops Manager deploy

Key Pair Name*

SSH Private Key*

Region*

Encrypt EBS Volumes

Save

3. Select **Use AWS Keys** or **Use AWS Instance Profile**.

- If you choose to use AWS keys, complete the fields with information from the **Outputs** tab for your stack in the AWS Console:

- **Access Key ID:** Use the value of **PcfIamUserAccessKey**.
- **AWS Secret Key:** Use the value of **PcfIamUserSecretAccessKey**.

- 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:** Use the value of **PcfVpc** from your **Outputs** tab.
- **Security Group ID:** Open the AWS EC2 Dashboard and click **Security Groups**. Select the security group with the **Description** **PCF VMs Security Group**. Copy the **Group ID** of this group into the Ops Manager **Security Group ID** field.
- **Key Pair Name:** Use the name of your pre-existing AWS key pair. You selected this key pair name when you first [deployed Ops Manager Director](#).
- **SSH Private Key:** Open your AWS key pair **.pem** file in a text editor. 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 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 on using EBS encryption.

5. Click **Save**.

Step 4: Director Config Page

1. Select **Director Config** to open the **Director Config** page.

The screenshot shows the 'Director Config' page with the following fields and settings:

- NTP Servers (comma delimited)*:** A text input field containing "0.amazon.pool.ntp.org, 1.amazon.pool.ntp.org".
- Metrics IP Address:** An empty text input field.
- Enable VM Resurrector Plugin:** A checkbox that is unchecked.
- Enable Post Deploy Scripts:** A checkbox that is unchecked.
- Recreate all VMs:** A checkbox that is unchecked. A note below it states: "This will force BOSH to recreate all VMs on the next deploy. Persistent disk will be preserved."
- Enable bosh deploy retries:** A checkbox that is unchecked. A note below it states: "This will attempt to re-deploy a failed deployment up to 5 times."
- Keep Unreachable Director VMs:** A checkbox that is unchecked.

2. In the **NTP Servers (comma delimited)** field, enter at least two of the following NTP servers, 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. (Optional) Enter your **Metrics IP Address** if you are [Using JMX Bridge](#).

4. Select the **Enable VM Resurrector Plugin** checkbox to enable the Ops Manager Resurrector functionality and increase Elastic Runtime availability.

5. 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.

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

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

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

9. Select **HM Pager Duty Plugin** to enable Health Monitor integration with PagerDuty.

The screenshot shows the configuration for the HM Pager Duty Plugin:

- HM Pager Duty Plugin:** A checkbox that is checked.
- Service Key***: A text input field containing "YOUR-PAGERDUTY-SERVICE-KEY".
- HTTP Proxy**: A text input field containing "YOUR-HTTP-PROXY".

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

The screenshot shows a configuration form for the HM Email Plugin. The fields are as follows:

- Host***: smtp.example.com
- Port***: 25
- Domain***: cloudfoundry.example.com
- From***: user2@example.com
- Recipients***: user@example.com, user1@example.com
- Username**: user
- Password**: (Redacted)
- Enable TLS**: checked

10. 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.

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

Blobstore Location

- Internal
- S3 Compatible Blobstore

S3 Endpoint*

https://s3-us-west-2.amazonaws.com

Bucket Name*

pcf-test-bucket-1234567890abcdef

Access Key*

AKIAJGZCQWVDR4A6G5Q

Secret Key*

[Change](#)

V2 Signature

V4 Signature

Region*

- In a browser, reference the [Amazon Simple Storage Service \(Amazon S3\) table](#), and find the region for your AWS account.
 - 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.
 - Complete the following fields with information from the **Outputs** tab in the AWS Console:
 - **Bucket Name:** Use the value of **PcfOpsManagerS3Bucket**.
 - **Access Key ID:** Use the value of **PcfIamUserAccessKey**.
 - **AWS Secret Key:** Use the value of **PcfIamUserSecretAccessKey**.
 - Select **V2 Signature** or **V4 Signature**. If you select **V4 Signature**, enter your **Region**.
 - *AWS recommends using Signature Version 4 when possible.*

 Note: For more information about AWS S3 Signatures, see the AWS [Authenticating Requests](#) documentation.

12. For **Database Location**, select **External MySQL Database**. Complete the following fields with information from the **Outputs** tab in the

The screenshot shows a configuration form for an external MySQL database. The fields are as follows:

- Database Location:** External MySQL Database (radio button selected)
- Host:** [Redacted]
- Port:** 3306
- Username:** [Redacted]n
- Password:** [Redacted]
Change
- Database:** bosh
- Max Threads:** [Redacted]
- Director Hostname:** opsdirector.example.com

A blue **Save** button is at the bottom.

AWS Console.

- **Host:** Use the value of **PcfRdsAddress**.
- **Port:** Use the value of **PcfRdsPort**.
- **Username:** Use the value of **PcfRdsUsername**.
- **Password:** Use the value of **PcfRdsPassword**.
- **Database:** Use the value of **PcfRdsDBName**.

13. **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.

14. (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](#).

15. Click **Save**.

Note: For more information about AWS S3 Signatures, see the AWS [Authenticating Requests](#) documentation.

Step 5: Create Availability Zones Page

Note: Pivotal recommends at least three Availability Zones for a highly available installation of Elastic Runtime.

1. Select **Create Availability Zones**.

Create Availability Zones

Availability Zones

us-west-1c

Amazon Availability Zone*

us-west-1c

The Amazon Availability Zone name (ex: 'us-east-1b')

Add

Save

This screenshot shows a user interface for creating an availability zone. At the top, it says 'Create Availability Zones'. Below that, it lists an existing availability zone 'us-west-1c'. Underneath it, there's a field labeled 'Amazon Availability Zone*' containing the value 'us-west-1c'. A note next to the field says 'The Amazon Availability Zone name (ex: 'us-east-1b')'. At the bottom right, there's a small trash can icon, and at the bottom left, there are 'Add' and 'Save' buttons.

2. Use the following steps to create one or more Availability Zones for your applications to use:
 - Click **Add**.
 - For **Amazon Availability Zone**, enter the value of **PcfPrivateSubnetAvailabilityZone** from the **Outputs** tab in the AWS Console.
 - **(Optional)** If you are using a second **Amazon Availability Zone**, click **Add**. Enter the value of **PcfPrivateSubnet2AvailabilityZone** from the **Outputs** tab in the AWS Console.
 - Click **Save**.

Step 6: Create Networks Page

1. Select **Create Networks**.

Create Networks

Warning: Pivotal recommends keeping the IP settings throughout the life of your installation. Ops Manager may prevent you from changing them in the future. Contact Pivotal support for help completing such a change.

Verification Settings

Enable ICMP checks

Networks

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

Deadmines [Add Network](#) 

Name* 

Service Network

Subnets

[Add Subnet](#)

VPC Subnet ID*

CIDR*

Reserved IP Ranges Ops Manager will not deploy VMs to any IP in this range, e.g. '10.9.9.0-10.9.9.100, 10.9.9.200-10.9.9.255'

DNS* One or more Domain Name Servers used by VMs

Gateway*

Availability Zones* AZ1

[Save](#)

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.



Note: To use the [Single Sign-On for PCF](#) service, you must configure a network with only one subnet.

- In the **VPC Subnet ID** field, use the value of **PcfPrivateSubnetId** from the **Outputs** tab in the AWS Console.
- For **CIDR**, enter `10.0.16.0/20`. Ops Manager deploys VMs to this CIDR block.
- For **Reserved IP Ranges**, enter `10.0.16.1-10.0.16.9`. Ops Manager avoids deploying VMs to any IP address in this range.
- Enter `10.0.0.2` for **DNS** and `10.0.16.1` for **Gateway**.
- Select which **Availability Zones** to use with the network.
- **(Optional)** If you are using a second subnet, click **Add Subnet**. In the **VPC Subnet ID** field, use the value of **PcfPrivateSubnet2Id**. Enter the rest of the fields using the information provided above.



Note: If you are using multiple Availability Zones, you must add a new network with at least one subnet for each Availability Zone.

4. Click **Save**.

5. If the following ICMP error message appears, you can ignore the warning. Dismiss the warning, and move on to the next step.

The screenshot shows a red alert box on the PCF Ops Manager interface. It contains the following text:

⚠ Please review the errors below

- Cannot reach gateway with IP 10.0.16.1 (ignorable if ICMP is disabled)
- Cannot reach DNS with IP 10.0.0.2 (ignorable if ICMP is disabled)

All errors will be reverified before installation.

Step 7: Assign AZs and Networks Page

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

The screenshot shows the 'Assign AZs and Networks' page. The title is 'Assign AZs and Networks'. The page instructions state: 'The Ops Manager Director is a single instance. Choose the availability zone in which to place that instance. It is highly recommended that you backup this VM on a regular basis to preserve settings.' Below this, there are two dropdown menus: 'Singleton Availability Zone' set to 'AZ1' and 'Network' set to 'Deadmines'. A large blue 'Save' button is at the bottom.

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 8: Security Page

The screenshot shows the 'Security' page with the following interface:

- Trusted Certificates:** A text area containing a redacted certificate snippet starting with "-----BEGIN CERTIFICATE-----".
- VM Password Generation:** A section with two radio button options:
 - Generate passwords
 - Use default BOSH password
- Save Button:** A blue 'Save' button at the bottom.

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 Trusted Registries for running app instances in Docker containers, use this field to enter your certificate for your private Docker Trusted Registry. See the [Using Docker Trusted 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: Resource Config Page

1. Select **Resource Config**.

The screenshot shows the 'Resource Config' page with the following table:

JOB	INSTANCES	PERSISTENT DISK TYPE	VM TYPE
Ops Manager Director	Automatic: 1	Automatic: 50 GB	Automatic: medium.disk (cpu: 2, ram: 4 GB)
Master Compilation Job	Automatic: 4	None	Automatic: large.cpu (cpu: 4, ram: 4 GB, di)

Save button is located at the bottom left.

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.

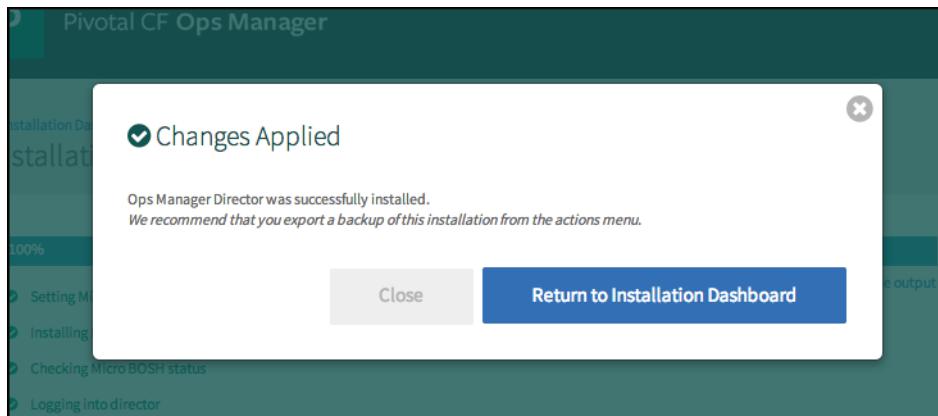
Note: If you set a field to **Automatic** and the recommended resource allocation changes in a future version, Ops Manager automatically uses the updated recommended allocation.

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 Elastic Runtime on AWS CloudFormation](#) topic.

Deploying Elastic Runtime on AWS

Page last updated:

This topic describes how to install and configure Elastic Runtime after deploying the CloudFormation template for Pivotal Cloud Foundry (PCF) on Amazon Web Services (AWS). Use this topic when [installing Pivotal Cloud Foundry on AWS](#).

Before beginning this procedure, ensure that you have successfully completed all steps in the [Deploying the CloudFormation Template for PCF on AWS](#) and [Configuring Ops Manager Director after Deploying PCF on AWS using CloudFormation](#) topics.

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 Elastic Runtime tile.

Step 1: Open the Outputs Tab in AWS

1. In the dashboard of your [AWS Console](#), click **CloudFormation**. The Stacks Dashboard displays.
2. Select the **pcf-stack** checkbox, then select the **Outputs** tab.

Key	Value	Description
PcfPublicSubnetAvailabilityZone	us-west-1b	
PcfPrivateSubnetAvailabilityZone	us-west-1b	
PcfPrivateSubnet2AvailabilityZone	us-west-1c	
PcfElasticRuntimeS3DropletsBucket	pcf-stack-felasticruntimes3dropletsbucket-1lb9ij48t4v29	
PcfElbSshDnsName	pcf-stack-f-ssh-elb-1859745201.us-west-1.elb.amazonaws.com	
PcfVmsSecurityGroupId	sg-043a5c60	
PcfElasticRuntimeS3BuildpacksBucket	pcf-stack-felasticruntimes3buildpacksbucket-kbx64zosvzs	
PcfVpc	vpc-36715a53	
PcfPublicSubnetId	subnet-100e3e49	
PcfRdsPassword	bosh1234	
PcfIamUserName	pcf-stack-psManStack-1AGSRAPW2HVO5-PcfiamUser-IC0JC0UZOW00	
PcfPrivateSubnet2Id	subnet-a79e74c3	
PcfRdsDBName	bosh	
PcfOpsManagerS3Bucket	pcf-stack-osmanstack-1agsrap-pcfopsmanagers3bucket-kypy81fqlas	
PcfPublicSubnetId2	subnet-a09e74c4	
PcfElasticRuntimeS3DropletsBucket2	pcf-stack-felasticruntimes3dropletsbucket-1nccfbhuu	

In the steps described below, use the information from the **Value** column of the **Outputs** tab to configure your PCF installation.

Step 2: Add Elastic Runtime to Ops Manager

1. Navigate to the Pivotal Cloud Foundry Operations Manager Installation Dashboard.
2. If you have not downloaded Elastic Runtime, click the Pivotal Network link on the left to download the Elastic Runtime .pivotal file. Click **Import a Product** to add the tile to Ops Manager. For more information, refer to the [Adding and Deleting Products](#) topic.
3. Click the **Elastic Runtime** tile in the Installation Dashboard.

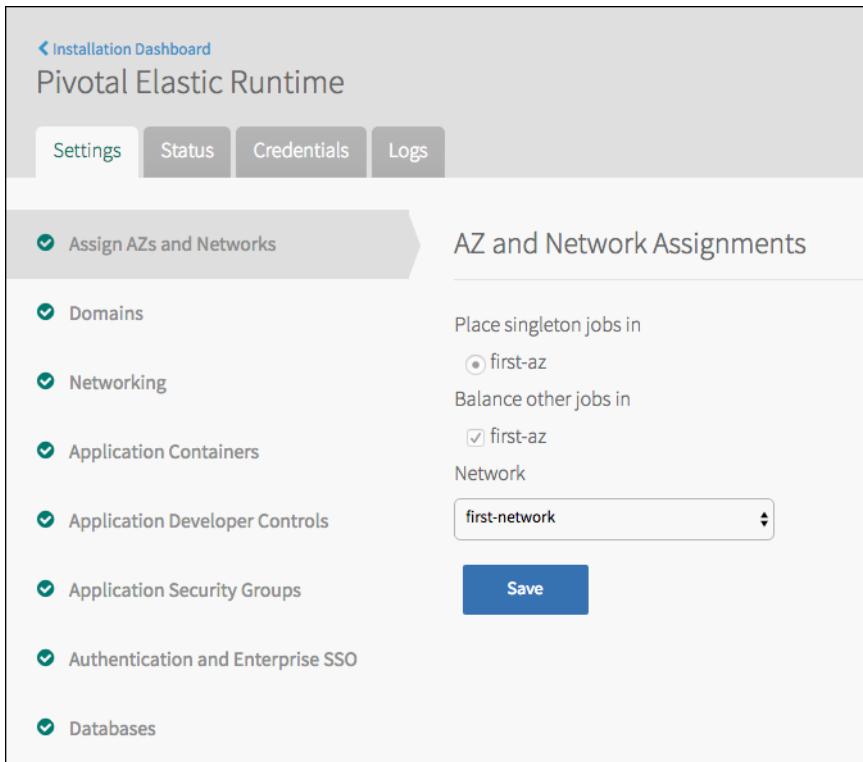


Step 3: Assign Availability Zones and Networks

1. Select **Assign AZ and Networks**. These are the Availability Zones that you [create](#) when configuring Ops Manager Director.
2. Select an 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.

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

4. From the **Network** drop-down box, choose the network on which you want to run Elastic Runtime.



Installation Dashboard

Pivotal Elastic Runtime

Settings Status Credentials Logs

Assign AZs and Networks

Domains

Networking

Application Containers

Application Developer Controls

Application Security Groups

Authentication and Enterprise SSO

Databases

AZ and Network Assignments

Place singleton jobs in

first-az

Balance other jobs in

first-az

Network

first-network

Save

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 4: Add CNAME Record for Your Custom Domain

In the [Use the AWS CLI to upload your SSL Cert](#) step, you uploaded an SSL certificate for your PCF wildcard domain to AWS. In this step you redirect all wildcard queries for your domain to the DNS name of your ELB.

Note: Do not point your wildcard domain at the numeric IP address for your ELB because this changes frequently.

- Find the DNS hostname of your ELB. The **Output** tab of the CloudFormation page in the AWS dashboard lists this as the value for the key **PcfElbDnsName**.
- Log in to the DNS registrar that hosts your domain (for example, Network Solutions, GoDaddy, or Register.com).
- Create a CNAME record with your DNS registrar that points `*.YOUR-DOMAIN.com` to the DNS hostname of your ELB.
- Save changes within the web interface of your DNS registrar.
- In the terminal, run the following `dig` command to confirm that you created your CNAME record successfully:

```
dig xyz.MY-DOMAIN.COM
```

You should see the CNAME record that you just created:

```
; ANSWER SECTION:  
xyz.MY-DOMAIN.COM. 1767 IN CNAME pcf-ert-frankfurt-pcf-elb-428333773.eu-central-1.elb.amazonaws.com.
```

Note: You **must** complete this step before proceeding to Cloud Controller configuration. A problem that is difficult to resolve can occur if the wildcard domain is improperly cached before the CNAME is registered.

Step 5: Configure Domains

- Select **Domains**.

Elastic Runtime hosts applications at subdomains under its apps domain and assigns system components to subdomains under its system domain. You need to configure a wildcard DNS for both the apps domain and system domain. The two domains can be the same, although this is not recommended.

System Domain *	<input type="text" value="system.example.com"/>
This domain is for system-level PCF components, such as Apps Manager, service brokers, etc. You must set up a wildcard DNS record for this domain that points to your entry point load balancer or HAProxy.	
Apps Domain *	<input type="text" value="apps.example.com"/>
Save	

- Enter the system and application domains.
 - The **System Domain** defines your target when you push apps to Elastic Runtime.
 - The **Apps Domain** defines where Elastic Runtime should serve your apps.

Note: Pivotal recommends that you use the same domain name but different subdomain names for your system and app domains.

This prevents system and apps routes from overlapping. You will require two wildcard DNS entries: one for the system and the other for apps. For example, `*.system.EXAMPLE.COM` and `*.apps.EXAMPLE.COM`. Point both wildcard domains at your internal router IP address, which can be found under the status tab in the Elastic Runtime tile.

 **Note:** You configured a wildcard DNS record for these domains in an earlier step.

- Click **Save**.

Step 6: Configure Networking

- Select **Networking**.
- 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 Elastic Runtime later in the installation process. See the [Configure Router to Elastic Load Balancer](#) section of this topic for more information.

- Under **Configure the point-of-entry to this environment** choose one of the following options:

 **Note:** For details about the different SSL/TLS termination point options, how they correspond to different points-of-entry for Elastic Runtime, and related certificate requirements, see the [Providing a Certificate for your SSL Termination Point](#) topic.

- Forward SSL to Elastic Runtime Router:** Select this option if your deployment uses an external load balancer that can forward encrypted traffic to the Elastic Runtime Router, or for a development environment that does not require load balancing. Complete the fields for the **Router SSL Termination Certificate and Private Key** and **Router SSL Ciphers**.

Select one of the following point-of-entry options:^{*}

Forward SSL to Elastic Runtime Router. Assumes an external load balancer is configured to forward encrypted traffic.

Router SSL Termination Certificate and Private Key ^{*}



Change
Required for SSL encryption between the load balancer and router(s).

Router SSL Ciphers

Forward unencrypted traffic to Elastic Runtime Router. Assumes an external load balancer is configured to forward unencrypted traffic.
 Forward SSL to HA Proxy. Like first option - Assumes an external load balancer is configured to forward encrypted traffic.

- Forward unencrypted traffic to Elastic Runtime Router:** Select this option if your deployment uses an external load balancer that cannot forward encrypted traffic to the Elastic Runtime Router, or for a development environment that does not require load balancing.

Select one of the following point-of-entry options:^{*}

Forward SSL to Elastic Runtime Router. Assumes an external load balancer is configured to forward encrypted traffic.
 Forward unencrypted traffic to Elastic Runtime Router. Assumes an external load balancer is configured to forward unencrypted traffic.
 Forward SSL to HA Proxy. Like first option - Assumes an external load balancer is configured to forward encrypted traffic.

Disable SSL certificate verification for this environment

Disable insecure cookies on the Router

- Forward SSL to HAProxy:** Select this option to use HAProxy as your first point of entry. Complete the fields for **SSL Certificate and Private Key** and **SSL Ciphers**.

Private Key, and HAProxy SSL Ciphers. Select **Disable HTTP traffic to HAProxy** if you want the HAProxy to only allow HTTPS traffic. For more information, on configuring HAProxy, see [Configuring SSL/TLS Termination at HAProxy](#).

Select one of the following point-of-entry options:*

- Forward SSL to Elastic Runtime Router. Assumes an external load balancer is configured to forward encrypted traffic.
- Forward unencrypted traffic to Elastic Runtime Router. Assumes an external load balancer is configured to forward unencrypted traffic.
- Forward SSL to HA Proxy. Like first option - Assumes an external load balancer is configured to forward encrypted traffic.

SSL Certificate and Private Key *

```
R7uE...[REDACTED]-----[REDACTED]
-----BEGIN RSA PRIVATE KEY-----
MIIE...[REDACTED]-----[REDACTED]
```

```
-----BEGIN RSA PRIVATE KEY-----
MIIE...[REDACTED]-----[REDACTED]
```

[Generate RSA Certificate](#)

Disable HTTP traffic to HAProxy

HAProxy SSL Ciphers

Request Max Buffer Size *

16384

Disable SSL certificate verification for this environment

4. 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.

5. Select the **Disable insecure cookies on the Router** checkbox to set the secure flag for cookies generated by the router.
 6. 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](#).

- Disable SSL certificate verification for this environment
- Disable insecure cookies on the Router
- Enable Zipkin tracing headers on the router

- 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.
 - 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`.

9. (Optional) Use the **Applications Subnet** field if you need to avoid address collision with a third-party service on the same subnet as your apps. Enter a CIDR subnet mask specifying the range of available IP addresses assigned to your app containers. The IP range must be different from the network used by the system VMs.

10. (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 . Some configurations, such as networks that use GRE tunnels, may require a smaller MTU value.

Choose whether to enable route services. Route services enable you to proxy requests to your app over TLS to arbitrary URLs before hitting your app.*

Enable route services
 Disable route services

Loggregator Port

Applications Subnet (Only change this if you need to avoid address collision with a third-party service on the same subnet.) *

Applications Network Maximum Transmission Unit (MTU) (in bytes) *

11. (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.

12. (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.

Router Timeout to Backends (in seconds) (min: 1) *

Load Balancer Unhealthy Threshold *

Load Balancer Healthy Threshold *

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

Router Max Idle Keepalive Connections (min: 0, max: 50000) *

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

- Select the **Enable TCP Routing** radio button.
- 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 . Configuration of this field is only applied on the first deploy, and update updates to the port range are made using the of CLI. For details about modifying the port range, see the [Router Groups](#) topic.

Enable TCP requests to your apps via specific ports on the TCP router. You will want to configure a load balancer to forward these TCP requests to the TCP routers. If you do not have a load balancer, then you can also send traffic directly to the TCP router.*

- Select this option if you prefer to enable TCP Routing at a later time
- Enable TCP Routing

TCP Routing Ports (one-time configuration, if you want to update this value you can via the CF CLI) *

1024-1123

- 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 Elastic Runtime. For more information, see the [Configure Router to Elastic Load Balancer](#) topic.

15. To disable TCP routing, select the **Select this option if you prefer to enable TCP Routing at a later time** radio button. For more information, see the [Configuring TCP Routing in Elastic Runtime](#) topic.
16. Click **Save**.

Step 7: Configure Application Containers

1. Select **Application Containers**.

Enable microservice frameworks, private Docker registries, and other services that support your applications at a container level.

Enable Custom Buildpacks

Allow SSH access to app containers

Enable SSH when an app is created

Private Docker Insecure Registry Whitelist

10.10.10.10:8888,example.com:8888



Docker Images Disk-Cleanup Scheduling on Cell VMs*

- Never clean up Cell disk-space
- Routinely clean up Cell disk-space
- Clean up disk-space once threshold is reached

Threshold of Disk-Used (MB) (min: 1) *

Max Inflight Container Starts *

Save

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 Elastic Runtime 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 Trusted Registries](#) topic for more information.
6. Select your preference for **Docker Images Disk-Cleanup on Cell VMs**. If you choose **Clean up disk-space once threshold is reached** enter a **Threshold of Disk-Used** in megabytes.
7. Optionally, 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. Click **Save**.

Step 8: Configure Application Developer Controls

1. Select **Application Developer Controls**.

Configure restrictions and default settings for applications pushed to Elastic Runtime.

Maximum File Upload Size (MB) (min: 1024, max: 2048) *	<input type="text" value="1024"/>
Default App Memory (MB) (min: 64, max: 2048) *	<input type="text" value="1024"/>
Default App Memory Quota per Org (min: 10240, max: 102400) *	<input type="text" value="10240"/>
Maximum Disk Quota per App (MB) (min: 512, max: 10240) *	<input type="text" value="2048"/>
Default Disk Quota per App (MB) (min: 512, max: 10240) *	<input type="text" value="1024"/>
Default Service Instances Quota per Org (min: 0, max: 1000) *	<input type="text" value="100"/>
<input type="button" value="Save"/>	

2. Enter your intended maximum file upload size.
3. Enter your default RAM memory allocation per app.
4. Enter your default total RAM memory (RAM) quota per Org. You can change this in the CLI.
5. Enter your maximum and default disk quotas per app.
6. Enter your default service instances quota per Org. You can change this in the CLI.
7. Click **Save**.

Step 9: Review Application Security Groups

Setting appropriate [Application Security Groups](#) is critical for a secure deployment. Type in the box to acknowledge that once the Elastic Runtime deployment completes, you will review and set the appropriate application security groups.

Setting appropriate Application Security Groups that control application network policy is the responsibility of the Elastic Runtime administration team. Please refer to the Application Security Groups topic in the Pivotal Cloud Foundry documentation for more detail on completing this activity after the Elastic Runtime deployment completes.

Type X to acknowledge that you understand this message *

Save

Step 10: Configure Authentication and Enterprise SSO

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

Configure your user store access, which can be an internal user store (managed by Cloud Foundry's UAA) or an external user store (LDAP or SAML). You can also adjust the lifetimes of authentication tokens.

Configure your UAA user account store with either internal or external authentication mechanisms*

- Internal UAA (provided by Elastic Runtime; configure your password policy below)

Minimum Password Length *

Minimum Uppercase Characters Required for Password *

Minimum Lowercase Characters Required for Password *

Minimum Numerical Digits Required for Password *

Minimum Special Characters Required for Password *

Number of Months Before Password Expires *

Maximum Password Entry Attempts Allowed *

- SAML Identity Provider

- LDAP Server

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 Elastic Runtime](#) 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 Elastic Runtime](#) topic.

3. (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.
4. (Optional) Customize the text prompts used for username and password from the cf CLI and Apps Manager login popup.
5. (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 public IP address(es), append a regular expression or regular expressions to match the public IP address(es).

The screenshot shows a configuration form with the following fields:

- Apps Manager Access Token Lifetime (in seconds)**: Value: 1209600
- Apps Manager Refresh Token Lifetime (in seconds)**: Value: 1209600
- Cloud Foundry CLI Access Token Lifetime (in seconds)**: Value: 7200
- Cloud Foundry CLI Refresh Token Lifetime (in seconds)**: Value: 1209600. A tooltip indicates: "Set the lifetime of the refresh token for the Cloud Foundry CLI."
- Customize Username Label (on login page)**: Value: Email
- Customize Password Label (on login page)**: Value: Password
- Proxy IPs Regular Expression**: Value: 10\.\d{1,3}\.\d{1,3}\.\d{1,3}|192\.168\.\d{1,3}\.\d{1,3}

A blue **Save** button is at the bottom.

6. Click **Save**.

Step 11: Configure System Databases

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

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**.

Place the databases used by Elastic Runtime components like Cloud Controller and UAA.

Choose the location of your system databases*

- Internal Databases - MySQL and Postgres (the Postgres DBs are not highly-available, but this selection is required if you want to keep your system data from a pre-1.6.0 Elastic Runtime that you upgraded)
- Internal Databases - MySQL (preferred for complete high-availability)
- External Databases (preferred if, for example, you use AWS RDS)

Save

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 Elastic Runtime databases, you must first create the databases on the RDS instance provided by the CloudFormation script.

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:

```
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 your AWS dashboard [Outputs tab](#) to fill in the following keys:

```
mysql --host=PcfRdsAddress --user=PcfRdsUsername --password=PcfRdsPassword
```

4. Run the following MySQL commands to create databases for the seven Elastic Runtime components that require a relational database:

```
CREATE database uaa;
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;
```

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

6. In Elastic Runtime, select **Databases**.

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

Place the databases used by Elastic Runtime components like Cloud Controller and UAA.

Choose the location of your system databases*

- Internal Databases - MySQL and Postgres (the Postgres DBs are not highly-available, but this selection is required if you want to keep your system data from a pre-1.6.0 Elastic Runtime that you upgraded)
- Internal Databases - MySQL (preferred for complete high-availability)
- External Databases (preferred if, for example, you use AWS RDS)

Hostname *

TCP Port *

App Usage Service database username *

App Usage Service database password *

8. For the **Hostname** and **TCP Port** fields, enter the corresponding values from the [Outputs tab](#) in the AWS Console, according to the following table:

Elastic Runtime Field	Outputs Key
Hostname	PcfRdsAddress
TCP Port	PcfRdsPort

9. For each **database username** and **database password** field, enter the corresponding values from the [Outputs tab](#) in the AWS Console, according to the following table:

Elastic Runtime Field	Outputs Key
DATABASE-NAME database username	PcfRdsUsername
DATABASE-NAME database password	PcfRdsPassword

10. 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 [Proxy](#) section of the *MySQL for PCF* topic for more information.

Only configure this section if you selected Internal Databases - MySQL or Internal Databases - MySQL and Postgres in the previous Databases section.

A proxy tier routes MySQL connections from internal components to healthy cluster nodes. Configure DNS and/or your own load balancer to point to multiple proxy instances for increased availability. TCP healthchecks can be configured against port 1936.

The automated backups functionality works with any S3-compatible file store that can receive your backup files.

MySQL Proxy IPs

MySQL Service Hostname

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.
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.

Replication canary time period *
<input type="text" value="30"/>
Replication canary read delay *
<input type="text" value="20"/>
E-mail address (required) *
<input type="text" value="email@example.com"/>

7. Under **Automated Backups Configuration**, choose one of three options for MySQL backups:
 - **Disable automatic backups of MySQL**
 - **Enable automated backups from MySQL to an S3 bucket or other S3-compatible file stores** saves your backups to an existing Amazon Web Services (AWS) or [Ceph](#) S3-compatible blobstore.

Automated Backups Configuration*

Disable automated backups of MySQL

Enable automated backups from MySQL to an S3 bucket or other S3-compatible file store

S3 Endpoint URL *

S3 Bucket Name *

Bucket Path *

AWS Access Key ID *

AWS Secret Access Key *

Cron Schedule *

Backup All Nodes

Enable automated backups from MySQL to a remote host via SCP

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 **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 automated backups from MySQL to a remote host via SCP** saves your backups to a remote host using secure copy

Automated Backups Configuration*

Disable automated backups of MySQL

Enable automated backups from MySQL to an S3 bucket or other S3-compatible file store

Enable automated backups from MySQL to a remote host via SCP

Hostname *

Port *

Username *

Private key *

Destination directory *

Cron Schedule *

Backup All Nodes

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 Elastic Runtime tile to 0.

8. 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.

The screenshot shows a configuration panel with the following fields:

- Server Activity Logging***
 - Disable server activity logging
 - Enable server activity logging
- Event types ***
- Load Balancer Healthy Threshold ***
- Load Balancer Unhealthy Threshold ***
- Save** button

9. 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.

10. Click **Save**.

Step 13: Configure File Storage

To minimize system downtime, Pivotal recommends using highly resilient and redundant *external* filestores for your Elastic Runtime file storage.

When configuring file storage for the Cloud Controller in Elastic Runtime, 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 Elastic Runtime 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 Elastic Runtime file storage, perform the following steps:

1. In the Elastic Runtime tile, select **File Storage**.

This section determines where you would like to place your Elastic Runtime Cloud Controller's file storage.

Configure your Cloud Controller's filesystem*

- Internal WebDAV (provided by Elastic Runtime)
- External S3-Compatible File Store (if you want to use a service like S3 or Ceph)

URL Endpoint*

`https://s3.amazonaws.com`

Access Key *

`XYZ1234567`

Secret Key *

`.....`

S3 Signature Version*

V4 Signature

Region*

US West (N. California)

- Server-side Encryption
(available for AWS S3 only)

Buildpacks Bucket Name*

`PcfElasticRuntimeS3Buildpack`

Droplets Bucket Name*

`PcfElasticRuntimeS3DropletBu`

Packages Bucket Name*

`PcfElasticRuntimeS3Packages`

Resources Bucket Name*

`PcfElasticRuntimeS3Resources`

- External Google Cloud Storage
- External Azure Storage

Save

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.
- Use the values in your AWS [Outputs tab](#) to complete the remaining fields as follows:

Ops Manager Field	Outputs Key
Buildpacks Bucket Name	PcfElasticRuntimeS3BuildpacksBucket
Droplets Bucket Name	PcfElasticRuntimeS3DropletsBucket
Packages Bucket Name	PcfElasticRuntimeS3PackagesBucket
Resources Bucket Name	PcfElasticRuntimeS3ResourcesBucket
Access Key ID	PcfIamUserAccessKey
AWS Secret Key	PcfIamUserSecretAccessKey

 **Note:** For more information regarding AWS S3 Signatures, see the [Authenticating Requests](#) documentation.

- 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 14: (Optional) Configure System Logging

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

- Select **System Logging**.

Configure system logging. Complete the External Syslog fields only if using an external syslogd server.

Enable Cloud Controller security event logging

External Syslog Aggregator Hostname
 

External Syslog Aggregator Port
 The typical syslog port is 514. Ensure syslogd is listening on external interfaces.

External Syslog Network Protocol
 

Syslog Drain Buffer Size (# of messages) *

Save

- 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).
- 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 Elastic Runtime 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**.

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](#) and [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.

Customize colors, images, and text for Apps Manager and the Cloud Foundry login portal.

Company Name



Accent Color

Main Logo (PNGs only)

Square Logo/Favicon (PNGs only)

Footer Text

Defaults to 'Pivotal Software Inc. All rights reserved.'

Add

Footer Links

You may configure up to three links in the Apps Manager footer

Classification Header/Footer Background Color

Classification Header/Footer Text Color

Classification Header Content

Classification Footer Content

Save

2. Click **Save** to save your settings in this section.
3. Select **Apps Manager**. Use this section to configure page names and sidebar links in the **Apps Manager** and **Marketplace** pages.

Configure Apps Manager

Display Marketplace Service Plan Prices

Supported currencies as json *



Define the currency codes and associated symbols (defaults to "[\"usd\": \"\$\", \"eur\": \"€\"]" if blank)

Product Name

Marketplace Name

Customize Sidebar Links
You may configure up to 10 links in the Apps Manager sidebar 

Link	Remove
▶ Marketplace	
▶ Docs	
▶ Tools	

Save

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

Step 16: (Optional) Configure Email Notifications

Elastic Runtime 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**.

Configure Simple Mail Transfer Protocol for the Notifications application to send email notifications about your deployment. This application is deployed as an errand in Elastic Runtime. If you do not need this service, leave this section blank and disable the Notifications and Notifications UI errands.

From Email

Address of SMTP Server

Port of SMTP Server

SMTP Server Credentials

[Change](#)

SMTP Enable Automatic STARTTLS

SMTP Authentication Mechanism*

SMTP CRAMMD5 secret

Save

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: (Optional) Add CCDB Restore Key

Perform this step if all of the following are true:

- You deployed Elastic Runtime previously.
- You then stopped Elastic Runtime or it crashed.
- You are re-deploying Elastic Runtime with a backup of your Cloud Controller database.

1. Click **Restore CCDB Encryption Key**.
2. Enter your **Cloud Controller DB Encryption Key**.

You must enter your original database encryption key here once you have deployed Elastic Runtime for the first time. Anytime thereafter your encryption key will remain the same, if you specify the same key here. This is important for backup and restore of data as well as upgrades.

Cloud Controller DB Encryption Key

...

Leave blank unless restoring a Cloud Controller Database exported from a previous installation.

Save

See [Backing Up Pivotal Cloud Foundry](#) for more information.

Step 18: Configure Smoke Tests

The Smoke Tests errand runs basic functionality tests against your Elastic Runtime 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 org and space**, which creates an ad-hoc org and space for running smoke tests and deletes them afterwards. Otherwise, select **Specified org and space** and complete the fields to specify where you want to run smoke tests.

Specify a Cloud Foundry organization and space where smoke tests can run if in the future you delete your Elastic Runtime deployment domains.

Choose whether to create a new org and space on demand for smoke tests*

- Temporary org and space (This org and space are deleted after smoke tests finish.)
 Specified org and space (The org and space must have a domain available for routing.)

Organization *

Space *

Domain *

Save

3. Click **Save**.

Step 19: (Optional) Enable Advanced Features

The Advanced Features section of Elastic Runtime 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

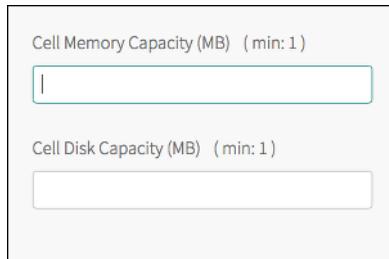
If your apps do not use the full allocation of disk space and memory set in the **Resource Config** tab, you might want use this feature. These fields control the amount to overcommit disk and memory resources to each Diego Cell VM.

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**.



The screenshot shows a form with two input fields. The first field is labeled "Cell Memory Capacity (MB) (min: 1)" with a text input box containing a single vertical bar character. The second field is labeled "Cell Disk Capacity (MB) (min: 1)" with an empty text input box.

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

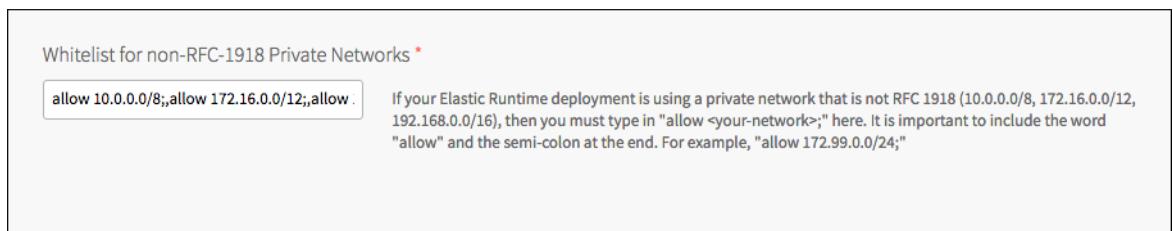
Some private networks require extra configuration so that internal file storage (WebDAV) can communicate with other PCF processes.

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.



The screenshot shows a text input field for the whitelist. It contains the text "allow 10.0.0.0/8;allow 172.16.0.0/12;allow ". To the right of the input field, there is a note: "If your Elastic Runtime deployment is using a private network that is not RFC 1918 (10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16), then you must type in "allow <your-network>;" here. It is important to include the word "allow" and the semi-colon at the end. For example, "allow 172.99.0.0/24;"

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 Elastic Runtime 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 Elastic Runtime.

To modify the value of the **CF CLI Connection Timeout**, perform the following steps:

1. Select **Advanced Features**.



A screenshot of a web interface showing a single input field with a light gray border. The field contains the number "15". Above the field, the text "CF CLI Connection Timeout" is displayed in a small, dark font.

2. Add a value, in seconds, to the **CF CLI Connection Timeout** field.
3. Click **Save**.

Container-to-Container Networking

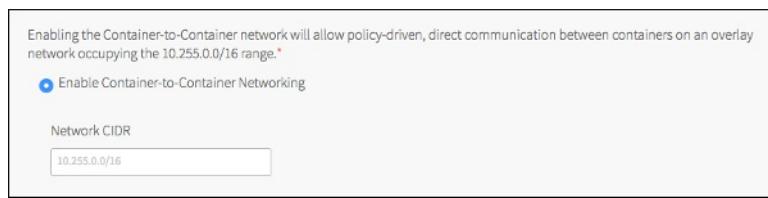
Enabling Container-to-Container Networking allows direct, policy-driven communication between containers on an overlay network occupying a specific range. For more information about Container-to-Container Networking, see the [Administering Cloud Foundry Networking](#) topic.

By default, PCF sets the overlay range to `10.255.0.0/16`, but you can modify this range when you enable Container to Container Networking.

Enable Container-to-Container Networking

To enable Container-to-Container Networking, perform the following steps:

1. Select **Advanced Features**.
2. Select **Enable Container-to-Container Networking**. Optionally, add a range of IP addresses in CIDR format to use instead of the default



A screenshot of a configuration dialog box. At the top, there is a note: "Enabling the Container-to-Container network will allow policy-driven, direct communication between containers on an overlay network occupying the 10.255.0.0/16 range." Below this, there is a radio button labeled "Enable Container-to-Container Networking" which is selected. Underneath, there is a section titled "Network CIDR" with an input field containing the value "10.255.0.0/16".

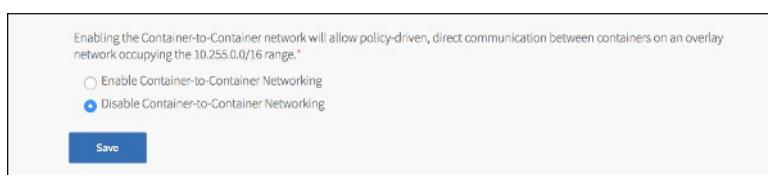
overlay network.

3. Click **Save**.

Disable Container-to-Container Networking

To disable Container-to-Container Networking, perform the following steps:

1. Select **Advanced Features**.
2. Select **Disable Container-to-Container Networking**.



A screenshot of a configuration dialog box. At the top, there is a note: "Enabling the Container-to-Container network will allow policy-driven, direct communication between containers on an overlay network occupying the 10.255.0.0/16 range." Below this, there are two radio buttons: "Enable Container-to-Container Networking" (unchecked) and "Disable Container-to-Container Networking" (checked). At the bottom of the dialog is a blue "Save" button.

3. Click **Save**.

CF API Rate Limiting

Enabling 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.

Enable CF API Rate Limiting

To enable CF API Rate Limiting, perform the following steps:

1. Select **Advanced Features**.
2. Under **Enable CF API Rate Limiting**, select **Enable**.

Enabling CF API Rate Limiting will prevent 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.*

Enable

General Limit *

2000

Unauthenticated Limit *

100

3. 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**.
4. For **Unauthenticated Limit**, enter the number of requests an unauthenticated client is allowed to make over an hour interval. The default value is **100**.
5. Click **Save**.

Disable CF API Rate Limiting

To disable CF API Rate Limiting, perform the following steps:

1. Select **Advanced Features**.
2. Under **Enable CF API Rate Limiting**, select **Disable**.

Enabling CF API Rate Limiting will prevent 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.*

Enable

Disable

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 Elastic Runtime. There are two types of errands: *post-deploy errands* run after the product is installed, and *pre-delete errands* run before the product is 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. The Elastic Runtime 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.

Errands

Errands are scripts that run at designated points during an installation.

Post-Deploy Errands

Smoke Test Errand

Runs Smoke Tests against your Elastic Runtime installation

On

Apps Manager Errand

Pushes the Pivotal Apps Manager application to your Elastic Runtime installation

On

Notifications Errand

Pushes the Pivotal Notifications application to your Elastic Runtime installation

On

Notifications UI Errand

Pushes the Notifications UI component to your Elastic Runtime installation

On

Pivotal Account Errand

Pushes the Pivotal Account application to your Elastic Runtime installation

On

Autoscaling Errand

Pushes the Pivotal App Autoscaling application to your Elastic Runtime installation

On

Autoscaling Registration Errand

Registers the Autoscaling Service Broker

On

NFS Broker Errand

Pushes the NFS Broker application to your Elastic Runtime installation

On

There are no pre-delete errands for this product.

Save

- **Smoke Test Errand** verifies that your deployment can do the following:
 - Push, scale, and delete apps
 - Create and delete orgs and spaces
- **Apps Manager Errand** deploys the Apps Manager, a dashboard for managing apps, services, orgs, users, and spaces. Until you deploy Apps Manager, you must perform these functions through the of CLI. After Apps Manager has been deployed, Pivotal recommends deselecting the checkbox for this errand on subsequent Elastic Runtime deployments. For more information about the 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 Elastic Runtime. See the [Enabling NFS Volume Services](#) topic for more information.

Step 21: Configure Router to Elastic Load Balancer

1. If you do not know it, find the name of your Elastic Load Balancer (ELB) by clicking **Load Balancers** in the AWS EC2 dashboard. This example shows three ELBs:

- `pcf-stack-pcf-ssh-elb` : An SSH load balancer
- `pcf-stack-pcf-elb` : A load balancer
- `pcf-stack-pcf-tcp-elb` : A TCP load balancer

<input type="checkbox"/>	Name	DNS name	State	VPC	Availability Zones
<input type="checkbox"/>	pcf-stack-pcf-ssh-elb	pcf-stack-pcf-ssh-elb-...	vpc-...	us-west-2a, us-west-2b	
<input type="checkbox"/>	pcf-stack-pcf-elb	pcf-stack-pcf-elb-172...	vpc-...	us-west-2a, us-west-2b	
<input type="checkbox"/>	pcf-stack-pcf-tcp-elb	pcf-stack-pcf-tcp-elb-...	vpc-...	us-west-2a, us-west-2b	

Resource Config				
JOB	INSTANCES	PERSISTENT DISK TYPE	VM TYPE	ELB NAMES
Consul	Automatic: 1	Automatic: 1 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)	
NATS	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)	
etcd	Automatic: 1	Automatic: 1 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)	
Diego BBS	Automatic: 1	Automatic: 1 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)	
File Storage	Automatic: 1	Automatic: 100 GB	Automatic: medium:mem (cpu: 1, ram: 8 GB, disk: 8 GB)	
MySQL Proxy	Automatic: 1	None	Automatic: small (cpu: 1, ram: 2 GB, disk: 4 GB)	
MySQL Server	Automatic: 1	Automatic: 100 GB	Automatic: large:disk (cpu: 2, ram: 8 GB, disk: 64 GB)	
Backup Prepare Node	0	Automatic: 200 GB	Automatic: small (cpu: 1, ram: 2 GB, disk: 4 GB)	
Cloud Controller Database (Postgres)	Automatic: 0	Automatic: 2 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)	
UAA Database (Postgres)	Automatic: 0	Automatic: 10 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)	
UAA	Automatic: 1	None	Automatic: medium:disk (cpu: 2, ram: 4 GB, disk: 32 GB)	
Cloud Controller	Automatic: 1	Automatic: 1 GB	Automatic: medium:disk (cpu: 2, ram: 4 GB, disk: 32 GB)	
HAProxy	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)	
Clock Global	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)	
Cloud Controller Worker	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)	
Collector	Automatic: 0	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)	
Diego Brain	Automatic: 1	Automatic: 1 GB	Automatic: small (cpu: 1, ram: 2 GB, disk: 4 GB)	
Diego Cell	Automatic: 3	None	Automatic: xlarge:disk (cpu: 4, ram: 16 GB, disk: 128 GB)	
Doppler Server	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)	
Loggregator Trafficcontroller	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)	
Router	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)	
TCP Router	0	Automatic: 1 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)	
Push Apps Manager	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)	
Run Smoke Tests	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)	
Push Notifications	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)	
Run Notifications Tests	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)	
Push Notifications UI	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)	
Run Notifications-UI Tests	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)	
Push AutoScaling	Automatic: 1	None	Automatic: nano (cpu: 1, ram: 512 MB, disk: 1 GB)	
Register AutoScaling Service Broker	Automatic: 1	None	Automatic: nano (cpu: 1, ram: 512 MB, disk: 1 GB)	
Destroy autoscaling service broker	Automatic: 1	None	Automatic: nano (cpu: 1, ram: 512 MB, disk: 1 GB)	
Run AutoScaling Tests	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)	
Run CF Acceptance Tests	Automatic: 1	None	Automatic: small (cpu: 1, ram: 2 GB, disk: 4 GB)	
Bootstrap	Automatic: 1	None	Automatic: small (cpu: 1, ram: 2 GB, disk: 4 GB)	
Push Pivotal Account	Automatic: 1	None	Automatic: small (cpu: 1, ram: 2 GB, disk: 4 GB)	

2. In the **Elastic Runtime** tile, click **Resource Config**.

3. In the **ELB Name** field of the **Diego Brain** row, enter the name of your SSH load balancer. Specify multiple load balancers by entering the names separated by commas.

Diego Brain	Automatic: 1	Automatic: 1 GB	Automatic: small (cpu: 1, ram: 2 GB, disk: 4 GB)	pcf-stack-pcf-ssh-elb
-------------	--------------	-----------------	--	-----------------------

4. In the **ELB Name** field of the **Router** row, enter the name of your load balancer. Specify multiple load balancers by entering the names separated by commas.

Router	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)	pcf-stack-pcf-elb
--------	--------------	------	--	-------------------

5. In the **ELB Name** field of the **TCP Router** row, enter the name of your TCP load balancer if you enabled TCP routing in the [Advanced Features](#) pane. Specify multiple load balancers by entering the names separated by commas.

TCP Router	0	Automatic: 1 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)	pcf-stack-pcf-tcp-elb
------------	---	-----------------	--	-----------------------

6. Click **Save**.

Step 22: (Optional) Disable Unused Resources

By default, Elastic Runtime uses an internal filestore and internal databases. If you configure Elastic Runtime to use external resources, you can disable the corresponding system-provided resources in Ops Manager to reduce costs and administrative overhead.

For more information regarding scaling instances, see the [Zero Downtime Deployment and Scaling in CF](#) and the [Scaling Instances in Elastic Runtime](#) topics.

Complete the following procedures to disable specific VMs in Ops Manager:

1. Click **Resource Config**.

2. If you configure Elastic Runtime to use an external S3-compatible filestore, edit the following fields:

- **File Storage:** Enter in **Instances**.
3. If you configure Elastic Runtime to use an external Relational Database Service (RDS), edit the following fields:
- **MySQL Proxy:** Enter in **Instances**.
 - **MySQL Server:** Enter in **Instances**.
 - **Cloud Controller Database:** Enter in **Instances**.
 - **UAA Database:** Enter in **Instances**.
4. If you are using an External Load Balancer instead of HAProxy, enter in the **Instances** field for **HAProxy**.
5. Click **Save**.

Step 23: Download Stemcell

This step is only required if your Ops Manager does not already have the Stemcell version required by Elastic Runtime.

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 file.

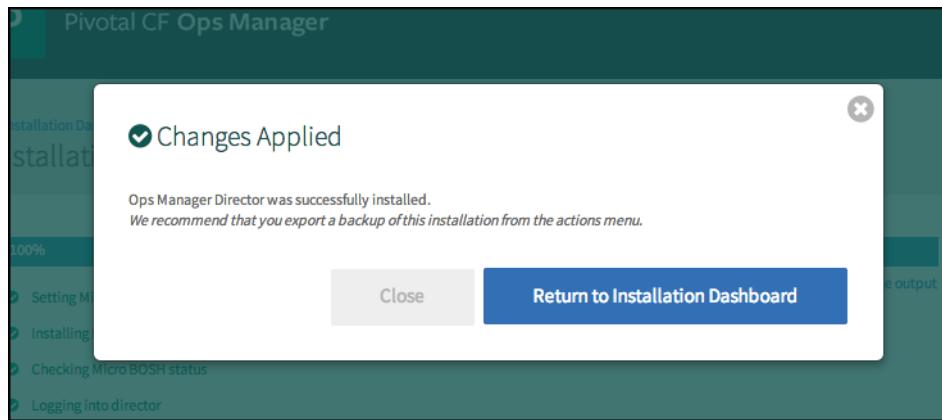
The screenshot shows a section titled "Stemcell". Below it is a description: "A stemcell is a template from which Ops Manager creates the VMs needed for a wide variety of components and products." Underneath, it says "cf requires BOSH stemcell version 3262 ubuntu-trusty" and provides a download link: "Using bosh-stemcell-3262.4-vsphere-esxi-ubuntu-trusty-go_agent.tgz". At the bottom is a large grey button labeled "Import Stemcell".

Step 24: Complete the Elastic Runtime 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 screenshot shows a red alert box with the text "⚠ Please review the errors below" and a bulleted list: "• Cannot reach gateway with IP 10.0.16.1 (ignorable if ICMP is disabled)" and "• Cannot reach DNS with IP 10.0.0.2 (ignorable if ICMP is disabled)". Below the alert are two buttons: "Ignore errors and start the install" (white background) and "Stop and fix errors" (red background).

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.

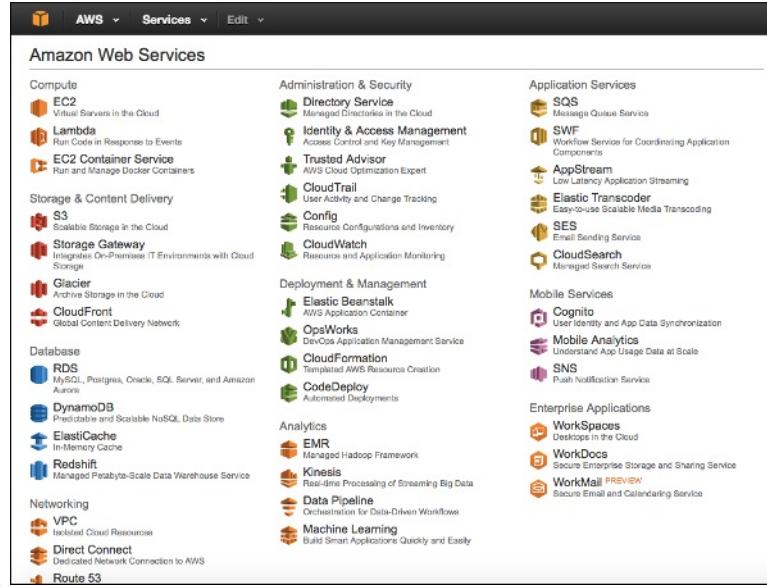


Return to [Installing Pivotal Cloud Foundry Using AWS CloudFormation](#).

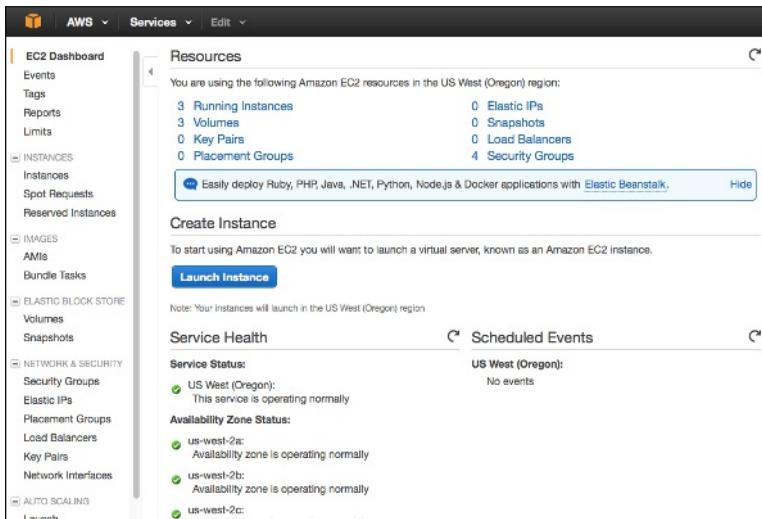
Deleting an AWS Installation from the Console

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.

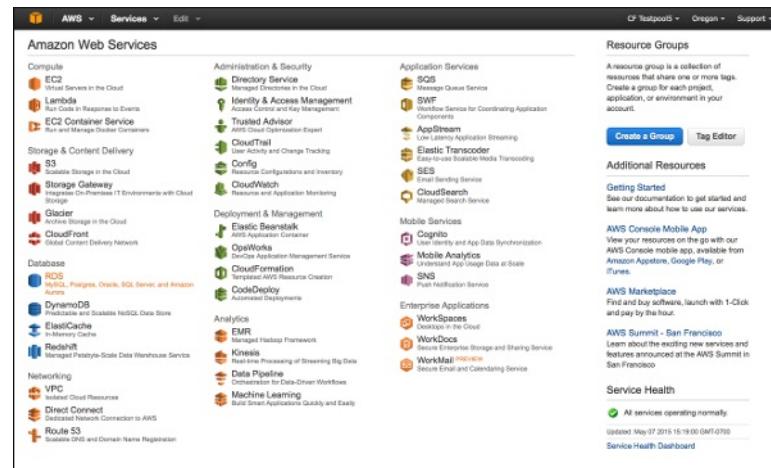


3. Terminate all your instances.

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with links like EC2 Dashboard, Events, Tags, Reports, Limits, Instances, Spot Requests, Reserved Instances, AMIs, Bundle Tasks, Volumes, Snapshots, Security Groups, Elastic IPs, Placement Groups, Load Balancers, Key Pairs, and Network Interfaces. The 'Instances' link is currently selected. In the main pane, three instances are listed: i-060963f0, i-0c0e64fa, and i-ba0e644c. Each instance has a green 'running' status indicator. An 'Actions' dropdown menu is open over the first instance, showing options: Connect, Get Windows Password, Launch More Like This, Instance State (with Start, Stop, Reboot, Terminate), Image, Networking, and CloudWatch Monitoring. The 'Terminate' option is highlighted with a red box. Below the instances, a table shows their descriptions: i-0c0e64fa: ec2-52-24-189-56.us-west-2.compute.amazonaws.com, i-060963f0: ec2-52-24-190-103.us-west-2.compute.amazonaws.com, and i-ba0e644c: ec2-52-24-189-159.us-west-2.compute.amazonaws.com.

The screenshot shows the AWS Load Balancers page. The sidebar is identical to the previous screenshot. In the main pane, there's a table with one row for a load balancer named 'CF-Test'. To the right of the table is a 'Actions' dropdown menu with options: Delete, Edit health check, Edit subnets, Edit instances, Edit listeners, and Edit security groups. The 'Delete' option is highlighted with a red box.

- Select **Load Balancers**. Delete all load balancers.



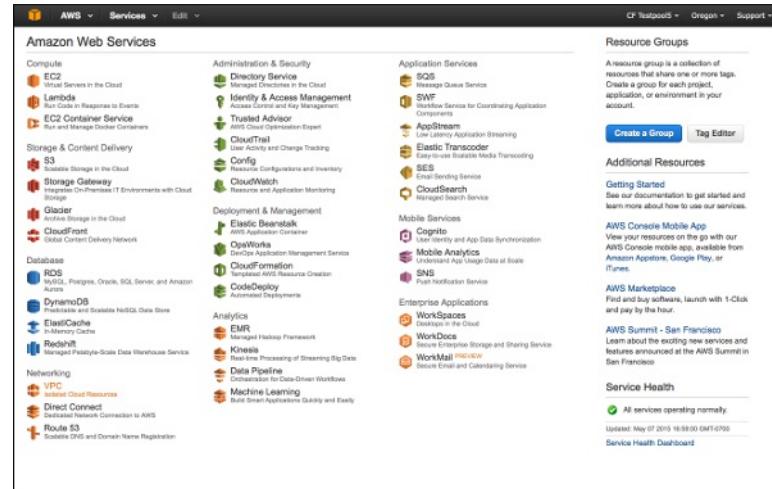
5. From the AWS Console, select **RDS**.

6. Select **Instances** from the menu on the left side. Delete the RDS instances.

The screenshot shows the RDS Dashboard with the Instances section selected. A context menu is open over a database instance named 'cf-test-db'. The menu options include 'Modify', 'Reboot', 'Delete', 'Create Read Replica', 'Promote Read Replica', 'Take DB Snapshot', 'Restore to Point in Time', 'See Details', and 'Subnets'. The 'Delete' option is highlighted in yellow.

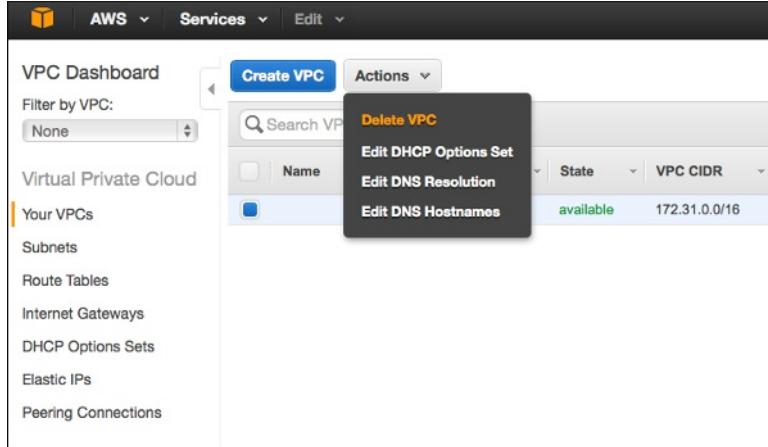
7. Select **Create final Snapshot** from the drop-down menu. Click **Delete**.

The screenshot shows the 'Delete DB Instance' dialog box. It asks if you want to delete the 'cf-test-db' instance. There are two options: 'Create final Snapshot?' (set to 'Yes') and 'Final snapshot name' (set to 'cf-test-db-fina-snapshot'). A warning message at the bottom says: 'We strongly recommend taking a final snapshot before instance deletion since after your instance is deleted, automated backups will no longer be available.' There are 'Cancel' and 'Delete' buttons at the bottom.

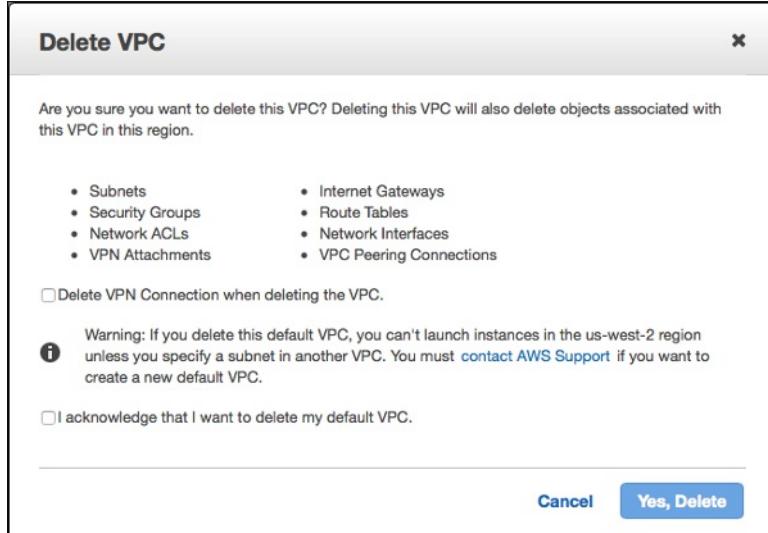


8. From the AWS Console, select **VPC**.

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**.



Guidelines for Creating User Roles on AWS

Pivotal recommends using the [CloudFormation templates for Pivotal Cloud Foundry](#) to configure AWS deployments to create users with least privilege. Pivotal also recommends minimizing the use of master account credentials by creating an IAM role and instance profile with the minimum required EC2, VPC, and EBS credentials.

Note: If you choose not to use the CloudFormation templates, Pivotal encourages you to use the permissions determined by [PcfIamPolicy](#) section of the Ops Manager CloudFormation template to create users with appropriate permissions. Additionally, follow AWS account security best practices such as disabling root keys, multi-factor authentication on the root account, and CloudTrail for auditing API actions.

See the table below for more information about the two CloudFormation templates.

Template Source	Location	User(s) Created	User Purpose	Uses IAM Role	Additional Documentation
Elastic Runtime	Pivotal Network Elastic Runtime Download	ERT S3 user	Blob storage	No	Deploying Elastic Runtime on AWS
Ops Manager	Referenced in the ERT template	Ops Manager VM and Ops Manager Director	EC2, VPC, EBS, S3, ELB	Yes	Director User Config

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 Amazon EBS Encryption

Page last updated:

Pivotal Cloud Foundry [\(PCF\)](#) supports [Amazon Elastic Block Store \(EBS\) Encryption](#) for PCF deployments on AWS. Amazon EBS Encryption allows operators to use full disk encryption for all persistent disks on BOSH-deployed VMs. You can use this feature to meet data-at-rest encryption requirements or as a security best practice.

There is no performance penalty for using encrypted EBS volumes. Pivotal advises all users of PCF on AWS to check this box.

How to Enable EBS Encryption

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



2. Select **AWS Config** to open the **AWS Management Console Config** page.

[Installation Dashboard](#)

Ops Manager Director

Settings Status Credentials

AWS Config Director Config Create Availability Zones Create Networks Assign AZs and Networks Security Resource Config

AWS Management Console Config

Use AWS Keys

Access Key ID*

AWS Secret Key*

Use AWS Instance Profile

AWS IAM Instance Profile*

VPC ID*

Security Group ID* The ID of the security group that will be assigned to your Ops Manager deploy

Key Pair Name*

SSH Private Key*

Region*

Encrypt EBS Volumes

Save

3. Select **Encrypt EBS Volumes**.

 **Note:** **Encrypt EBS Volumes** is a global setting. When selected, **Encrypt EBS Volumes** enables encryption on all VMs deployed by BOSH for all product tiles.

4. Click **Save**, and then return to the **Installation Dashboard**.

5. In Op Manager, click **Apply Changes** and review any reported errors. The following error message lists jobs that cannot be encrypted due to unsupported instance types.

A Please review the errors below

- Cannot reach gateway with IP 10.0.16.1 (ignorable if ICMP is disabled)
- Cannot reach DNS with IP 10.0.0.2 (ignorable if ICMP is disabled)
- Job 'nats' uses instance type t2.micro which does not support encryption. It will remain unencrypted.
- Job 'consul_server' uses instance type t2.micro which does not support encryption. It will remain unencrypted.
- Job 'etcd_server' uses instance type t2.micro which does not support encryption. It will remain unencrypted.
- Job 'diego_etcd' uses instance type t2.micro which does not support encryption. It will remain unencrypted.
- Job 'router' uses instance type t2.micro which does not support encryption. It will remain unencrypted.
- Job 'health_manager' uses instance type t2.micro which does not support encryption. It will remain unencrypted.
- Job 'clock_global' uses instance type t2.micro which does not support encryption. It will remain unencrypted.
- Job 'cloud_controller_worker' uses instance type t2.micro which does not support encryption. It will remain unencrypted.
- Job 'uaa' uses instance type t2.micro which does not support encryption. It will remain unencrypted.
- Job 'diego_brain' uses instance type t2.small which does not support encryption. It will remain unencrypted.
- Job 'doppler' uses instance type t2.micro which does not support encryption. It will remain unencrypted.
- Job 'loggregator_trafficcontroller' uses instance type t2.micro which does not support encryption. It will remain unencrypted.
- Job 'push-apps-manager' uses instance type t2.micro which does not support encryption. It will remain unencrypted.
- Job 'push-app-usage-service' uses instance type t2.micro which does not support encryption. It will remain unencrypted.
- Job 'smoke-tests' uses instance type t2.micro which does not support encryption. It will remain unencrypted.
- Job 'notifications' uses instance type t2.micro which does not support encryption. It will remain unencrypted.
- Job 'notifications-ui' uses instance type t2.micro which does not support encryption. It will remain unencrypted.
- Job 'autoscaling' uses instance type t2.micro which does not support encryption. It will remain unencrypted.
- Job 'autoscaling-register-broker' uses instance type t2.micro which does not support encryption. It will remain unencrypted.
- Job 'autoscaling-destroy-broker' uses instance type t2.micro which does not support encryption. It will remain unencrypted.
- Job 'diego-acceptance-tests' uses instance type t2.micro which does not support encryption. It will remain unencrypted.
- Job 'acceptance-tests' uses instance type t2.micro which does not support encryption. It will remain unencrypted.
- Job 'acceptance-tests-internetless' uses instance type t2.micro which does not support encryption. It will remain unencrypted.

[Ignore errors and start the install](#)

[Stop and fix errors](#)

If you find a job that should be encrypted in the error list, modify the instance type for that job in the **Resource Config** page of the Elastic Runtime. Select an instance type that supports encryption. Pivotal recommends using `t2.large`.

6. After you make your changes in Elastic Runtime, return to Ops Manager and click **Apply Changes**.

The next BOSH deploy encrypts all persistent disks on all BOSH-deployed VMs. If you have already deployed VMs with unencrypted EBS volumes, BOSH copies over all the data on those unencrypted EBS volumes to new encrypted volumes and discards the old volumes.

If you deselect **Encrypt EBS Volumes** later and then redeploy, BOSH overwrites all EBS volumes with unencrypted volumes.

Limitations

Using EBS Encryption is subject to the following limitations:

- Ops Manager and Director VMs are not encrypted.
- PCF does not support Amazon EBS Encryption for the following AWS instance types:
 - `t2.micro`
 - `t2.small`
 - `t2.medium`



Note: PCF will remove this limitation in a future release.

- Ephemeral disks are not encrypted. The **Encrypt EBS Volumes** checkbox applies only to persistent disks.
- Compilation worker VMs are not encrypted because they do not have persistent disks.

Creating a Proxy ELB for Diego SSH without CloudFormation

Page last updated:

If you want to allow SSH connections to application containers, you may want to use an Elastic Load Balancer (ELB) as the SSH proxy.

Users who deploy a [Pivotal Cloud Foundry](#) (PCF) 1.6+ installation on Amazon Web Services (AWS) using the CloudFormation template will automatically have this ELB created for them. However, if you are not using the CloudFormation template, or you are upgrading from an earlier version of PCF, perform the following steps to create this ELB in AWS manually:

1. On the EC2 Dashboard, click **Load Balancers**.
2. Click **Create Load Balancer**, and configure a load balancer with the following information:

Step 1: Define Load Balancer

Basic Configuration

This wizard will walk you through setting up a new load balancer. Begin by giving your new load balancer a unique name so that you can identify it from other load balancers you might create. You will also need to configure ports and protocols for your load balancer. Traffic from your clients can be routed from any load balancer port to any port on your EC2 instances. By default, we've configured your load balancer with a standard web server on port 80.

Load Balancer name:	my-SSH-ELB		
Create LB Inside:	vpc-4b38852e (10.0.0.0/16) pcf-vpc		
Create an internal load balancer:	<input type="checkbox"/> (what's this?)		
Enable advanced VPC configuration:	<input checked="" type="checkbox"/>		
Listener Configuration:			
Load Balancer Protocol	Load Balancer Port	Instance Protocol	Instance Port
TCP	2222	TCP	2222
Add			

Select Subnets

You will need to select a Subnet for each Availability Zone where you wish traffic to be routed by your load balancer. If you have instances in only one Availability Zone, please select at least two Subnets in different Availability Zones to provide higher availability for your load balancer.

VPC vpc-4b38852e (10.0.0.0/16) | pcf-vpc

A Please select at least two Subnets in different Availability Zones to provide higher availability for your load balancer.

Available Subnets				
Actions	Availability Zone	Subnet ID	Subnet CIDR	Name
+	us-west-1b	subnet-b63be1ef	10.0.16.0/20	pcf-private-subnet
+	us-west-1b	subnet-b73be1ee	10.0.2.0/24	pcf-rds-subnet-1
+	us-west-1c	subnet-c8f095ad	10.0.3.0/24	pcf-rds-subnet-2

Selected Subnets				
Actions	Availability Zone	Subnet ID	Subnet CIDR	Name
-	us-west-1b	subnet-b43be1ed	10.0.0.0/24	pcf-public-subnet

- 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**.

Create Security Group

Security group name: PCF_SSH_ELB_SecurityGroup

Description:

VPC: vpc-4b38852e (10.0.0.0/16) |pcf-vpc| * denotes default VPC

Security group rules:

Inbound Outbound

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

Add Rule

Create

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.

1. Define Load Balancer 2. Assign Security Groups **3. Configure Security Settings** 4. Configure Health Check 5. Add EC2 Instances 6. Add Tags 7. Review

Step 3: Configure Security Settings

⚠ Improve your load balancer's security. Your load balancer is not using any secure listener.

If your traffic to the load balancer needs to be secure, use either the HTTPS or the SSL protocol for your front-end connection. You can go back to the first step to add/configure secure listeners under [Basic Configuration](#) section. You can also continue with current settings.

7. Click **Next: Configure Health Check**.

1. Define Load Balancer 2. Assign Security Groups **3. Configure Security Settings**

Step 4: Configure Health Check

Your load balancer will automatically perform health checks on your EC2 instances. Instances removed from the load balancer will be removed from the load balancer. Customize the health check to meet your needs.

Ping Protocol	TCP
Ping Port	2222

Advanced Details

Response Timeout	5	seconds
Health Check Interval	30	seconds
Unhealthy Threshold	2	
Healthy Threshold	10	

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.

Create Record Set

Name: ssh.your-system-domain.com.

Type: CNAME – Canonical name

Alias: Yes No

TTL (Seconds): 300 | 1m | 5m | 1h | 1d

Value: your-elb-domain

The domain name that you want to resolve to instead of the value in the Name field.

Example:
www.example.com

14. You can now use this ELB to the SSH Proxy of your Elastic Runtime installation.
15. In Elastic Runtime, select **Resource Config**, and enter the ELB that you just created in the **Diego Brain** row, under the ELB Names column.

UAA	1	None	Automatic: t2.small (cpu: 1, ram: 2 GB, disk: 2 GB)	
Diego Brain	1	Automatic: 1 GB	Automatic: m3.medium (cpu: 1, ram: 3.75 GB, disk: 4 GE)	passion-elb
Diego Cell	3	None	m3.2xlarge (cpu: 8, ram: 30 GB, disk: 64 GB)	

Installing Pivotal Cloud Foundry on Azure

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](#).

Prerequisites

The following sections describe general requirements for running PCF and specific requirements for running PCF on Azure.

General Requirements

The following are general requirements for deploying and managing a PCF deployment with Ops Manager and Elastic Runtime:

- (**Recommended**) Ability to create a wildcard DNS record to point to your router or load balancer. Alternatively, you can use a service such as xip.io. For example, `203.0.113.0.xip.io`.
- (**Recommended**) A network without DHCP available for deploying the Elastic Runtime VMs

 **Note:** If you have DHCP, refer to the Troubleshooting Guide to avoid issues with your installation.

- Sufficient IP allocation:
 - One IP address for each VM instance
 - An additional IP address for each instance that requires static IPs
 - An additional IP address for each errand
 - An additional IP address for each compilation worker: `IPs needed = VM instances + static IPs + errands + compilation workers`
- The most recent version of the [Cloud Foundry Command Line Interface \(cf CLI\)](#)
- One or more NTP servers if not already provided by your IaaS

Azure Requirements

The following are the minimum resource requirements for maintaining a PCF deployment with Ops Manager and Elastic Runtime on Azure:

- VMs:
 - 27 F1s
 - 4 F2s
 - 4 F4s
 - 1 DS11v2
 - 1 DS12v2
- An OS disk of 120 GB for the Ops Manager VM

 **Note:** The F-series is not supported in North Central US, East Asia, Japan East, Australia East, and Australia Southeast. The DSv2-series is not supported in East Asia. If you are deploying PCF in one of these regions, see the [Ops Manager API documentation](#) for instructions on how to override the default VM sizes. Changing the default VM sizes may increase the cost of your deployment.

- An Azure subscription that can accommodate a public IP address for each VM

 **Note:** By default, Azure sets an upper limit of 60 public IP addresses. Depending on the size of your deployment, you may need to increase this limit by submitting an Azure support ticket. For more information, see the [Azure documentation](#).

- To deploy PCF on Azure, you must have the Azure CLI v0.10.5 or higher. For instructions on how to install the Azure CLI for your operating system, see [Preparing to Deploy PCF on Azure](#).

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 Elastic Runtime 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.

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](#)

Preparing to Deploy PCF on 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. Use the `azure --version` command to verify you have installed Azure CLI v0.10.5 or higher.

```
$ azure --version
```

If you have the correct version of the Azure CLI installed, skip the next step.

2. To install a new or updated Azure CLI, follow the instructions for your operating system:

- **Mac OS X:** [Download](#) and run the **Mac OS X Azure CLI** installer.
- **Windows:** [Download](#) and run the **Windows Azure CLI** installer. Use the command line, not PowerShell, to run the Azure CLI.
- **Linux:**
 1. If not already installed, install Node.js and npm.
 2. [Download](#) the Linux Azure CLI tar file.
 3. Run `sudo npm install -g PATH-TO-TAR-FILE`.
 4. If you encounter the error `/usr/bin/env: node: No such file or directory` when running `azure` commands, run
`sudo ln -s /usr/bin/nodejs /usr/bin/node`.

3. Set the mode of the Azure CLI to Azure Resource Management:

```
$ azure config mode arm
```

4. Log in to your Azure account:

```
$ azure login --environment AzureCloud
```

 **Note:** To target the Azure China environment, 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.

Step 2: Set Your Default Subscription

1. Run `azure account list --json` to list your Azure subscriptions:

```
$ azure account list --json
[
{
  "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"
}
]
```

- 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 `azure account set SUBSCRIPTION_ID` to set a new default, where `SUBSCRIPTION_ID` is the value of the **id** field. For example: `"87654321-1234-5678-1234-567891234567"`.

```
$ azure account set SUBSCRIPTION_ID
info: Executing command account set
info: Setting subscription to "Sample Subscription" with id "SUBSCRIPTION-ID".
info: Changes saved
info: account set command OK
```

- Record the value of the **id** set as the default. You use this value in future configuration steps.
- 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

- 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.

```
$ azure ad app create --name "Service Principal for BOSH" \
--password "PASSWORD" --home-page "http://BOSHAzureCPI" \
--identifier-uris "http://BOSHAzureCPI"
```

 **Note:** You can provide any string for the `home-page` and `identifier-uris` flags, but the value of `identifier-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.

- Record the value of **AppId** from the output. This is your `APPLICATION_ID` for creating a service principal.

```
info: Executing command ad app create
+ Creating application Service Principal for BOSH
data: AppId: 246e4af7-75b5-494a-89b5-363addb9f0fa
data: ObjectId: 208096bb-4899-49c2-83ea-1a270154f427
data: DisplayName: Service Principal for BOSH
data: IdentifierUris: 0=http://BOSHAzureCPI
data: ReplyUrls:
data: AvailableToOtherTenants: False
info: ad app create command OK
```

Step 4: Create and Configure a Service Principal

- To create a service principal, run `azure ad sp create --applicationId YOUR-APPLICATION-ID`, replacing `YOUR-APPLICATION-ID` with the `APPLICATION_ID` you recorded in the [previous step](#):

```
$ azure ad sp create --applicationId YOUR-APPLICATION-ID
info: Executing command ad sp create
+ Creating service principal for application YOUR-APPLICATION-ID
data: Object Id: fcf68d7a-262b-42c4-8ef8-6a4856611155
data: Display Name: Service Principal for BOSH
data: Service Principal Names:
data:     YOUR-APPLICATION-ID
data:     http://BOSHAzureCPI
info: ad sp create command OK
```

- You must have the Contributor role on your service principal to deploy PCF. Run the following command to assign this role:

```
$ azure role assignment create --spn "SERVICE-PRINCIPAL-NAME"\ \
--roleName "Contributor" --subscription 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 [RBAC: Built-in roles](#).

- Verify the assignment by running the following command:

```
$ azure role assignment list --spn "SERVICE-PRINCIPAL-NAME"

info: Executing command role assignment list
+ Searching for role assignments
data: RoleAssignmentId : /subscriptions/112a3bbc-44de-56ff-a7b8-9a012bbc3456/providers/Microsoft.Authorization/roleAssignments/061581af-118b-45e9-95a5-4e4ccf22c75d
data: RoleDefinitionName : Contributor
data: RoleDefinitionId : b24988ac-6180-42a0-ab88-20f7382dd24c
data: Scope : /subscriptions/112a3bbc-44de-56ff-a7b8-9a012bbc3456
data: DisplayName : Service Principal for BOSH
data: SignInName : undefined
data: ObjectId : 11b11a1-11c1-1111-a222-3df3f33e333f
data: ObjectType : ServicePrincipal
data:
info: role assignment list command OK
```

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`. Replace `YOUR-ENVIRONMENT` with `AzureCloud` or `AzureChinaCloud`.

```
$ azure login --username APPLICATION_ID --password CLIENT_SECRET \
--service-principal --tenant TENANT_ID --environment YOUR-ENVIRONMENT
info: Executing command login
- info: Added subscription Example
+
info: login command OK
```

If you cannot log in, the service principal is invalid. Create a new service principal and try again.

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.

Launching an Ops Manager Director Instance with an ARM Template

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. For more information, see the [Launching an Ops Manager Director Instance on Azure without an ARM Template](#) topic.

Before you perform the procedures in this topic, you must complete the procedures in the [Preparing to Deploy PCF on Azure](#) topic. After you complete the procedures in this topic, follow the instructions in [Configuring Ops Manager Director on Azure](#).

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 three 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 `azure location list`.

3. Create your resource group:

```
$ azure group create $RESOURCE_GROUP $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. Register your subscription with Microsoft.Storage:

```
$ azure provider register Microsoft.Storage
```

6. Create the storage account.

```
$ azure storage account create -l $LOCATION -g $RESOURCE_GROUP \
--sku-name LRS --kind Storage $STORAGE_NAME
```

7. Retrieve the connection string for your BOSH storage account:

```
$ azure storage account connectionstring show $STORAGE_NAME \
--resource-group $RESOURCE_GROUP
```

The command returns the following output:

```
info: Executing command storage account connectionstring show
+ Getting storage account keys
data: connectionstring: DefaultEndpointsProtocol=https;AccountName=example-storage;AccountKey=accountkeystring
info: storage account connectionstring show command OK
```

8. From the `data:` field in the output above, record the full value of `connectionstring`, starting with and including
`DefaultEndpointsProtocol=`.

9. Export the connection string:

```
$ export CONNECTION_STRING="YOUR-CONNECTION-STRING"
```

10. Create a container for the Ops Manager image:

```
$ azure storage container create opsman-image \
--connection-string $CONNECTION_STRING
```

11. Create a container for the Ops Manager VM:

```
$ azure storage container create vhds \
--connection-string $CONNECTION_STRING
```

12. Create a container for Ops Manager:

```
$ azure storage container create opsmanager \
--connection-string $CONNECTION_STRING
```

13. Create a container for BOSH:

```
$ azure storage container create bosh \
--connection-string $CONNECTION_STRING
```

14. Create a container for the stemcell:

```
$ azure storage container create stemcell --permission blob \
--connection-string $CONNECTION_STRING
```

 **Note:** Make sure the stemcell storage container is assigned `blob` permissions.

15. Create a table for stemcell data:

```
azure storage table create 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:

```
$ azure storage blob copy start $OPS_MAN_IMAGE_URL opsmanager \
--dest-connection-string $CONNECTION_STRING \
--dest-container opsman-image \
--dest-blob image.vhd
```

5. Copying the image may take several minutes. Run the following command and examine the output under `Status` to check the status:

```
$ azure storage blob copy show opsman-image image.vhd --connection-string $CONNECTION_STRING
info: Executing command storage blob copy show
+ Getting storage blob information
data: Copy ID          Progress      Status
data: -----
data: 069d413d-be05-4b12-82bc-c96dacee230e 31457280512/31457280512 success
info: storage blob copy show command OK
```

When `Status` reads `success`, continue to the next step.

Step 3: Configure the ARM Template

1. 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.

2. Download [the ARM template](#), `azure-deploy.json`.
3. Download [the parameters file](#), `azure-deploy-parameters.json`.
4. Change into the directory that contains `azure-deploy.json` and `azure-deploy-parameters.json`.
5. Open the parameters file and enter values for the following parameters:
 - storageAccountName**: The name of the storage account you created in [Step 1: Create Storage Account](#)
 - adminSSHKey**: The contents of the `opsman.pub` public key file that you created above
 - tenantID**: Your tenant ID, retrieved in the [Preparing to Deploy PCF on Azure](#) topic
 - clientID**: Your client or application ID, retrieved in the [Preparing to Deploy PCF on Azure](#) topic
 - clientSecret**: Your client secret, created in the [Preparing to Deploy PCF on Azure](#) topic
 - vmSize**: The size of the Ops Manager VM. Pivotal recommends using `Standard_DS2_v2`.
 - location**: The location where to install the Ops Manager VM. For example, `westus`.

Step 4: Deploy the ARM Template and Deployment Storage Accounts

1. Deploy the template:

```
$ azure group deployment create -f azure-deploy.json \
-e azure-deploy-parameters.json -v $RESOURCE_GROUP cfdeploy
```

2. When the command finishes, examine the last five lines of the output:

```
data: Name          Type   Value
data: -----
data: opsMan-FQDN    String  pcf-opsman-e8ddgelqlq22.westus.cloudapp.azure.com
data: extra Storage Account Prefix String  xtrastrgm7qcfdqj1q62
info: group deployment create command OK
```

Record the following values:

- opsMan-FQDN**: In the example above, `pcf-opsman-e8ddgelqlq22.westus.cloudapp.azure.com`
- extra Storage Account Prefix**: In the example above, `xtrastrgm7qcfdqj1q62`

3. The template creates three new deployment storage accounts. The names of the deployment storage accounts are the value of `extra Storage Account Prefix` appended with `1`, `2`, and `3`. In the example above, the names of the three deployment storage accounts are:
 - `xtrastrgm7qcfdqj1q621`
 - `xtrastrgm7qcfdqj1q622`
 - `xtrastrgm7qcfdqj1q623`

For each of the three new deployment storage accounts, perform the following steps:

1. Retrieve the connection string for your storage account, replacing `YOUR-DEPLOYMENT-STORAGE-ACCOUNT-NAME` with the name of the

storage account. For example, `xtrastrgm7qcfdq1jlq621`.

```
$ azure storage account connectionstring \
show YOUR-DEPLOYMENT-STORAGE-ACCOUNT-NAME \
--resource-group $RESOURCE_GROUP
```

The command returns output similar to the following:

```
info: Executing command storage account connectionstring show
+ Getting storage account keys
data: connectionstring: DefaultEndpointsProtocol=https;AccountName=example-storage;AccountKey=accountkeystring
info: storage account connectionstring show command OK
```

2. From the `data:` field in the output above, record the full value of `connectionstring` from the output above, starting with and including `DefaultEndpointsProtocol=`.
3. Export the connection string, choosing a unique name for `CONNECTION_STRING_N`. For example, `CONNECTION_STRING_2`.

```
$ export CONNECTION_STRING_N="YOUR-CONNECTION-STRING"
```

4. Create a container for Ops Manager:

```
$ azure storage container create opsmanager \
--connection-string $CONNECTION_STRING_N
```

5. Create a container for BOSH:

```
$ azure storage container create bosh \
--connection-string $CONNECTION_STRING_N
```

6. Create a container for the stemcell:

```
$ azure storage container create stemcell --permission blob \
--connection-string $CONNECTION_STRING_N
```



Note: Make sure the stemcell container is assigned `blob` permissions.

4. Create a network security group named `pcf-nsg`.

```
$ azure network nsg create $RESOURCE_GROUP pcf-nsg $LOCATION
```

5. Add a network security group rule to the `pcf-nsg` group to allow traffic from the public Internet.

```
$ azure network nsg rule create $RESOURCE_GROUP pcf-nsg internet-to-lb \
--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 `opsMan-FQDN` you retrieved from the output of the template deployment above.
2. Continue to the [Configuring Ops Manager Director on Azure](#) topic.

Launching an Ops Manager Director Instance on Azure without an ARM Template

Page last updated:

This topic describes how to deploy Ops Manager Director for Pivotal Cloud Foundry (PCF) on Azure by using individual commands to create resources in Azure instead of using an Azure Resource Manager (ARM) template. For information about using the ARM template, see the [Launching an Ops Manager Director Instance with an ARM Template](#) topic.

Before you perform the procedures in this topic, you must have completed the procedures in the [Preparing to Deploy PCF on Azure](#) topic. After you complete the procedures in this topic, follow the instructions in the [Configuring Ops Manager Director on Azure](#) 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="YOUR-LOCATION"
```

 **Note:** For a list of available locations, run `azurдеж location list`.

5. Create a network security group named `pcf-nsg`.

```
$ azure network nsg create $RESOURCE_GROUP pcf-nsg $LOCATION
```

6. Add a network security group rule to the `pcf-nsg` group to allow traffic from the public Internet.

```
$ azure network nsg rule create $RESOURCE_GROUP pcf-nsg internet-to-lb \
--protocol Tcp --priority 100 --destination-port-range '*'
```

 **Note:** Because the VMs do not have public IP addresses, this network security group rule only affects the load balancer.

7. Create a network security group named `opsmgr-nsg`.

```
$ azure network nsg create $RESOURCE_GROUP opsmgr-nsg $LOCATION
```

8. Add a network security group rule to the `opsmgr-nsg` group that allow HTTP traffic to the Ops Manager VM.

```
$ azure network nsg rule create $RESOURCE_GROUP opsmgr-nsg http \
--protocol Tcp --destination-port-range 80 --priority 100
```

9. Add a network security group rule to the `opsmgr-nsg` group that allow HTTPS traffic to the Ops Manager VM.

```
$ azure network nsg rule create $RESOURCE_GROUP opsmgr-nsg https \
--protocol Tcp --destination-port-range 443 --priority 200
```

- Add a network security group rule to the `opsmgr-msg` group that allow SSH traffic to the Ops Manager VM.

```
$ azure network nsg rule create $RESOURCE_GROUP opsmgr-nsg ssh \
--protocol Tcp --destination-port-range 22 --priority 300
```

- Create a virtual network named `pcf-net`.

```
$ azure network vnet create $RESOURCE_GROUP pcf-net $LOCATION \
--address-prefixes 10.0.0.0/16
```

- Add a subnet to the network for PCF VMs.

```
$ azure network vnet subnet create $RESOURCE_GROUP pcf-net pcf \
--address-prefix 10.0.0.0/20
```

 **Note:** To use the [Single Sign-On for PCF](#) service, you must configure a network that contains only one subnet.

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 three or more deployment storage accounts.

- 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"
```

- Create a BOSH storage account with the following command, replacing `SUBSCRIPTION_ID` with your subscription ID.

```
$ azure storage account create $STORAGE_NAME --resource-group $RESOURCE_GROUP \
--sku-name LRS --kind Storage --subscription SUBSCRIPTION_ID \
--location $LOCATION
```

If the command fails, ensure you have followed the rules for naming your storage account. Re-export a new storage account name if necessary.

- Configure the Azure CLI to use the BOSH storage account as its default.

- Retrieve the connection string for the account.

```
$ azure storage account connectionstring show $STORAGE_NAME \
--resource-group $RESOURCE_GROUP
```

The command returns output similar to the following:

```
info: Executing command storage account connectionstring show
+ Getting storage account keys
data: connectionstring: DefaultEndpointsProtocol=https;AccountName=example-storage;AccountKey=accountkeystring
info: storage account connectionstring show command OK
```

- From the `data:` field in the output above, record the full value of `connectionstring` from the output above, starting with and including `DefaultEndpointsProtocol=`.
- Export the value of `connectionstring` as the environment variable `$AZURE_STORAGE_CONNECTION_STRING`.

```
$ export AZURE_STORAGE_CONNECTION_STRING="YOUR-ACCOUNT-KEY-STRING"
```

- Create three blob containers in the BOSH storage account, named `opsmanager`, `bosh`, and `stemcell`.

```
$ azure storage container create opsmanager
$ azure storage container create bosh
$ azure storage container create stemcell --permission blob
```

- Create a table named `stemcells`.

```
$ azure storage table create stemcells
```

6. Choose a set of unique names for three 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`.

 **Note:** You can create up to 20 Azure storage accounts if your installation is large, or you can start with three and increase the number later.

7. Register your subscription with Microsoft.Storage:

```
$ azure provider register Microsoft.Storage
```

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.

```
$ azure storage account create MY_DEPLOYMENT_STORAGE_X \
--resource-group $RESOURCE_GROUP --sku-name LRS --kind Storage \
--subscription SUBSCRIPTION_ID --location $LOCATION
```

If the command fails, try a different set of account names.

- b. Retrieve the connection string for the account.

```
$ azure storage account connectionstring show MY_DEPLOYMENT_STORAGE_X \
--resource-group $RESOURCE_GROUP
```

The command returns output similar to the following:

```
info: Executing command storage account connectionstring show
+ Getting storage account keys
data: connectionstring: DefaultEndpointsProtocol=https;AccountName=example-storage;AccountKey=accountkeystring
info: storage account connectionstring show command OK
```

- c. From the `data:` field in the output above, record the full value of **connectionstring** from the output above, starting with and including `DefaultEndpointsProtocol=`.
d. Create three blob containers named `opsmanager`, `bosh`, and `stemcell` in the account.

```
$ azure storage container create opsmanager \
--connection-string "YOUR-ACCOUNT-KEY-STRING"
```

```
$ azure storage container create bosh \
--connection-string "YOUR-ACCOUNT-KEY-STRING"
```

```
$ azure storage container create stemcell \
--permission blob --connection-string "YOUR-ACCOUNT-KEY-STRING"
```

Step 3: Create a Load Balancer

1. Create a load balancer named `pcf-lb`.

```
$ azure network lb create $RESOURCE_GROUP pcf-lb $LOCATION
```

2. Create a static IP address for the load balancer named `pcf-lb-ip`.

```
$ azure network public-ip create $RESOURCE_GROUP pcf-lb-ip $LOCATION --allocation-method Static
info: Executing command network public-ip create
warn: Using default --idle-timeout 4
warn: Using default --ip-version IPv4
+ Looking up the public ip "pcf-lb-ip"
+ Creating public ip address "pcf-lb-ip"
data: Id : /subscriptions/222e8ffe-81ce-33ee-e3e2-1a405ffc4134/resourceGroups/pcf-resource-group/providers/Microsoft.Network/publicIPAddresses/pcf-lb-ip
data: Name : pcf-lb-ip
data: Type : Microsoft.Network/publicIPAddresses
data: Location : westus
data: Provisioning state : Succeeded
data: Allocation method : Static
data: IP version : IPv4
data: Idle timeout in minutes : 4
data: IP Address : 198.51.100.1
info: network public-ip create command OK
```

3. Record the **IP Address** 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.

```
$ azure network lb frontend-ip create $RESOURCE_GROUP pcf-lb pcf-fe-ip \
--public-ip-name pcf-lb-ip
```

5. Add a probe to the load balancer.

```
$ azure network lb probe create $RESOURCE_GROUP pcf-lb tcp80 \
--protocol Tcp --port 80
```

6. Add a backend address pool to the load balancer.

```
$ azure network lb address-pool create $RESOURCE_GROUP pcf-lb pcf-vms
```

 **Note:** This backend pool is empty when you create it.

7. Add a load balancing rule for HTTP.

```
$ azure network lb rule create $RESOURCE_GROUP pcf-lb http --protocol tcp \
--frontend-port 80 --backend-port 80
```

8. Add a load balancing rule for HTTPS.

```
$ azure network lb rule create $RESOURCE_GROUP pcf-lb https --protocol tcp \
--frontend-port 443 --backend-port 443
```

9. Add a load balancing rule for SSH.

```
$ azure network lb rule create $RESOURCE_GROUP pcf-lb diego-ssh --protocol tcp \
--frontend-port 2222 --backend-port 2222
```

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. Copy the Ops Manager image into your storage account.

```
$ azure storage blob copy start $OPS_MAN_IMAGE_URL opsmanager \
--dest-connection-string $AZURE_STORAGE_CONNECTION_STRING \
--dest-container opsmanager \
--dest-blob image.vhd
```

5. Copying the image may take several minutes. Run the following command and examine the output under `Status` to check the status:

```
$ azure storage blob copy show opsmanager image.vhd
info: Executing command storage blob copy show
+ Getting storage blob information
data: Copy ID          Progress      Status
data: -----
data: 069d413d-be05-4b12-82bc-c96dacee230e 31457280512/31457280512 success
info: storage blob copy show command OK
```

When `Status` reads `success`, continue to the next step.

6. Create a public IP address named `ops-manager-ip`.

```
$ azure network public-ip create $RESOURCE_GROUP ops-manager-ip $LOCATION --allocation-method Static
info: Executing command network public-ip create
warn: Using default --idle-timeout 4
warn: Using default --ip-version IPv4
+ Looking up the public ip "ops-manager-ip"
+ Creating public ip address "ops-manager-ip"
data: Id           : /subscriptions/222e8ffe-81ce-33ee-e3e2-1a405ffc4134/resourceGroups/pcf-resource-group/providers/Microsoft.Network/publicIPAddresses/ops-manager-ip
data: Name         : ops-manager-ip
data: Type         : Microsoft.Network/publicIPAddresses
data: Location     : westus
data: Provisioning state : Succeeded
data: Allocation method : Static
data: IP version    : IPv4
data: Idle timeout in minutes : 4
data: IP Address   : 192.0.2.1
info: network public-ip create command OK
```

7. Record the **IP Address** from the output above. This is the public IP address of Ops Manager.

8. Create a network interface for Ops Manager.

```
$ azure network nic create --subnet-vnet-name pcf-net --subnet-name pcf \
--network-security-group-name opsmgr-nsg \
--private-ip-address 10.0.0.5 --public-ip-name ops-manager-ip \
$RESOURCE_GROUP ops-manager-nic $LOCATION
```

 **Note:** If the command fails with a `parameters must be provided` error, run it again and ensure that you place `$RESOURCE_GROUP ops-manager-nic $LOCATION` at the end of the command as specified above. The Azure CLI requires that you specify options before other parameters.

9. 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.

10. Create a VM against the Ops Manager image, replacing `PATH-TO-PUBLIC-KEY` with the path to your public key `.pub` file.

```
$ azure vm create $RESOURCE_GROUP ops-manager $LOCATION \
Linux --nic-name ops-manager-nic \
--os-disk-vhd https://$STORAGE_NAME.blob.core.windows.net/opsmanager/os_disk.vhd \
--image-uri https://$STORAGE_NAME.blob.core.windows.net/opsmanager/image.vhd \
--admin-username ubuntu --storage-account-name $STORAGE_NAME \
--vm-size Standard_DS2_v2 --ssh-publickey-file PATH-TO-PUBLIC-KEY
```

11. The VM may take several minutes to boot. Run the following command, and examine the output under **PowerState** to check the status of

the VM.

```
$ azure vm list
info: Executing command vm list
+ Getting virtual machines
data: ResourceGroupName Name ProvisioningState PowerState Location Size
data: -----
data: PCF ops-manager Succeeded VM running westus Standard_DS2_v2
info: vm list command OK
```

The **PowerState** displays `VM running` when the Ops Manager deployment successfully completes.

Step 5: Resize Ops Manager VM Disk (Required)

The default OS disk size is too small for deploying PCF. Perform the following steps to increase the size of the OS disk of the Ops Manager VM:

1. Stop and deallocate the Ops Manager VM before resizing:

```
$ azure vm deallocate --resource-group $RESOURCE_GROUP --name ops-manager
```

2. Increase the size of the OS disk of the Ops Manager VM to 120 GB:

```
$ azure vm set --resource-group $RESOURCE_GROUP --name ops-manager --new-os-disk-size 120
```

3. Start the Ops Manager VM:

```
$ azure vm start --resource-group $RESOURCE_GROUP --name ops-manager
```

4. SSH into the Ops Manager VM, replacing `YOUR-OPS-MAN-IP` with the public IP address of Ops Manager that you recorded in a previous step:

```
$ ssh -i opsman YOUR-OPS-MAN-IP
```

If the private key you generated is not `opsman`, provide the correct filename instead.

5. From the Ops Manager VM, use `df` to confirm that the OS disk has been resized:

```
$ df -h
Filesystem  Size Used Avail Use% Mounted on
udev        1.7G  0   1.7G  0% /dev
tmpfs       344M 5.0M 340M  2% /run
/dev/sda1    119G 1.3G 48G  3% /
```

Step 6: 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 Director on Azure](#) topic.

Configuring Ops Manager Director on Azure

Page last updated:

This topic describes how to configure the Ops Manager Director for Pivotal Cloud Foundry (PCF) on Azure.

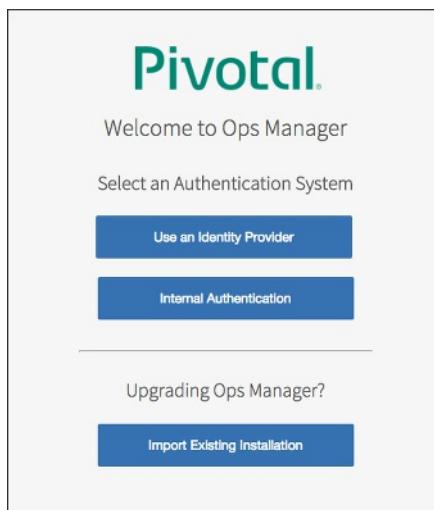
Before you perform the procedures in this topic, you must have completed the procedures in the [Preparing to Deploy PCF on Azure](#) topic, and either the [Launching an Ops Manager Director Instance with an ARM Template](#) topic or the [Launching an Ops Manager Director Instance on Azure without an ARM Template](#) topic.

After you complete the procedures in this topic, follow the instructions in the [Deploying Elastic Runtime on Azure](#) 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

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 are using 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**.

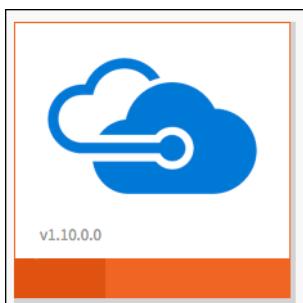
The screenshot shows the 'Internal Authentication' setup page. It includes fields for 'Username', 'Password', 'Password confirmation', 'Decryption passphrase', 'Decryption passphrase confirmation', and proxy settings ('Http proxy', 'Https proxy', 'No proxy'). A checkbox for agreeing to the terms and conditions is present, along with a link to the 'End User License Agreement'. A 'Setup Authentication' button is at the bottom.

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

The screenshot shows the 'Welcome!' sign-in page. It features fields for 'Email' and 'Password', and a 'SIGN IN' button. There is also a small horizontal line below the sign-in form.

Step 2: Azure Config Page

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



2. Select **Azure Config**.

[Installation Dashboard](#)

Ops Manager Director

Settings Status Credentials

Azure Config Director Config Create Networks Assign Networks Security Resource Config

Azure Config

Subscription ID* The Azure Subscription ID

Tenant ID*

Application ID*

Client Secret* Change

Resource Group Name* pcf

BOSH Storage Account Name* pcfstorage

Deployments Storage Account Name* *pcfdeploymentstorage*

Default Security Group* pcf-nsg

SSH Public Key*

SSH Private Key*

Save

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 `SRESOURCE_GROUP` environment variable.
- **BOSH Storage Account Name:** Enter the name of your storage account, which you exported as the `$STORAGE_NAME` environment variable.
- **Deployments Storage Account Name:** 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.

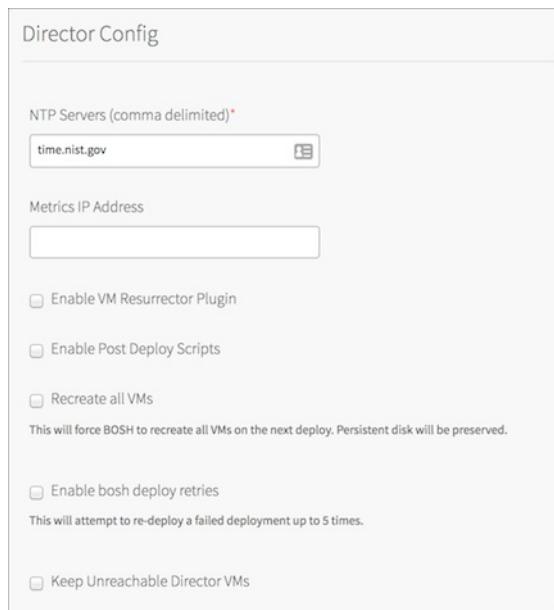
- **Default Security Group:** Enter `pcf-nsg`.

 **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.

5. 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.
6. For **SSH Private Key**, copy and paste the contents of your private key in the `opsman` file.
7. Click **Save**.

Step 3: Director Config Page

1. Select **Director Config**.



The screenshot shows the 'Director Config' configuration page. It includes fields for 'NTP Servers (comma delimited)' containing 'time.nist.gov', a 'Metrics IP Address' field with a placeholder, and several checkboxes for deployment behaviors: 'Enable VM Resurrector Plugin', 'Enable Post Deploy Scripts', 'Recreate all VMs' (with a note about preserving persistent disk), 'Enable bosh deploy retries' (with a note about up to 5 attempts), and 'Keep Unreachable Director VMs'.

2. In the **NTP Servers (comma delimited)** field, enter a comma-separated list of valid NTP servers.
3. (Optional) Enter your **Metrics IP Address** if you are [Using JMX Bridge](#).
4. Select the **Enable VM Resurrector Plugin** checkbox to enable the Ops Manager Resurrector functionality and increase Elastic Runtime availability.
5. 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.
6. Select **Recreate all VMs** to force BOSH to recreate all VMs on the next deploy. This process does not destroy any persistent disk data.
7. Select **Enable bosh deploy retries** if you want Ops Manager to retry failed BOSH operations up to five times.
8. Select **Keep Unreachable Director VMs** if you want to preserve Ops Manager Director VMs after a failed deployment for troubleshooting purposes.
9. (Optional) Select **HM Pager Duty Plugin** to enable Health Monitor integration with PagerDuty.

HM Pager Duty Plugin

Service Key*

YOUR-PAGERDUTY-SERVICE-KEY

HTTP Proxy

YOUR-HTTP-PROXY

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

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

HM Email Plugin

Host*

smtp.example.com

Port*

25

Domain*

cloudfoundry.example.com

From*

user2@example.com

Recipients*

user@example.com, user1@example.com

Username

user

Password

Enable TLS

- **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.

Blobstore Location

Internal

S3 Compatible Blobstore

S3 Endpoint*

Bucket Name*

Access Key*

Secret Key*

V2 Signature

V4 Signature

Region*

11. For **Blobstore Location**, Pivotal recommends that you keep **Internal** selected.

Database Location

Internal

External MySQL Database

Host*

Port*

Username*

Password*

Database*

12. For **Database Location**, Pivotal recommends that you keep **Internal** selected.

13. **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.

14. (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](#).

15. Click **Save**.

Step 4: Create Networks Page

1. Select **Create Networks**.

Create Networks

Warning: Pivotal recommends keeping the IP settings throughout the life of your installation. Ops Manager may prevent you from changing them in the future. Contact Pivotal support for help completing such a change.

Verification Settings

Enable ICMP checks

Networks

[Add Network](#)

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

▼ default



Name*

default

A unique name for this network

Service Network

Subnets

Azure Network Name*

pcf-net/pcf

CIDR*

10.0.0.0/20

Reserved IP Ranges

10.0.0.1-10.0.0.9

DNS*

168.63.129.16

Gateway*

10.0.0.1

[Save](#)

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 default.

5. 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.

6. To create a subnet, complete the following fields:

- **Azure Network Name:** Enter pcf-net/pcf. 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 10.0.0.0/20.

- **Reserved IP Ranges:** Enter the first 9 IP addresses of the subnet. For example, 10.0.0.1-10.0.0.9.

- **DNS:** Enter `168.63.129.16`.
- **Gateway:** Enter the first IP address of the subnet. For example, `10.0.0.1`.

7. 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**.

The screenshot shows a "Network Assignments" page. It has a header "Network Assignments". Below it is a section labeled "Network" with a dropdown menu containing the option "default". At the bottom is a blue "Save" button.

2. Under **Network**, select the `default` network you created from the dropdown menu.

3. Click **Save**.

Step 6: Security Page

1. Select **Security**.

The screenshot shows a "Security" page. It has a header "Security". Below it is a section labeled "Trusted Certificates" containing a text input field with a large amount of certificate data. At the bottom is a blue "Save" button.

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 Trusted Registries for running app instances in Docker containers, use this field to enter the certificate for your private Docker Trusted Registry. See the [Using Docker Trusted 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: Resource Config Page

1. Select **Resource Config**.

Resource Config

JOB	INSTANCES	PERSISTENT DISK TYPE	VM TYPE	LOAD BALANCERS	INTERNET CONNECTED
Ops Manager Director	Automatic: 1	Automatic: 50 GB	Automatic: Standard_F2s (cpu: 2, ram: 4 G)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Master Compilation Job	Automatic: 4	None	Automatic: Standard_F4s (cpu: 4, ram: 8 G)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Save

2. Ensure that the **Internet Connected** checkboxes are selected for all jobs. This gives all VMs a public IP address that enables outbound Internet access.

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. Azure offers a managed [source NAT \(SNAT\) service](#). However, Pivotal does not recommend using this service because Elastic Runtime deployments may fail due to SNAT performance bottlenecks.

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: 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 8: 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. After you complete this procedure, follow the instructions in the [Configuring Elastic Runtime for Azure](#) topic.

Deploying Elastic Runtime on Azure

Page last updated:

This topic describes how to install and configure Elastic Runtime on Azure.

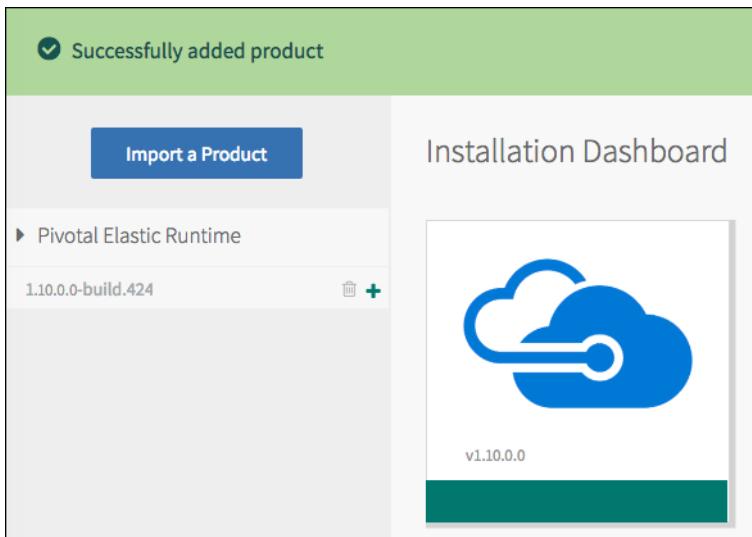
Before you perform the procedures in this topic, you must have completed the procedures in the [Preparing to Deploy PCF on Azure](#) topic, either the [Launching an Ops Manager Director Instance with an ARM Template](#) topic or the [Launching an Ops Manager Director on Azure without an ARM Template](#) 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 Elastic Runtime tile.

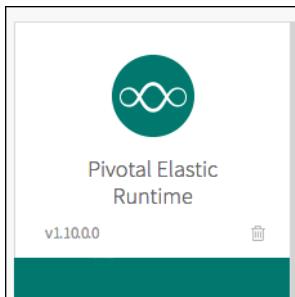
Note: The Azure portal sometimes displays the names of resources with incorrect capitalization. Always use the Azure CLI to retrieve the correctly capitalized name of a resource.

Step 1: Add Elastic Runtime to Ops Manager

1. Download Elastic Runtime from the [Pivotal Network](#).
2. Navigate to the Ops Manager Installation Dashboard.
3. Click **Import a Product** and select the downloaded `.pivotal` file. For more information, refer to the [Adding and Deleting Products](#) topic.
4. Click the plus button next to the imported tile to add it to the Installation Dashboard.



5. Click the **Elastic Runtime** tile in the Installation Dashboard.



Step 2: Assign Networks

1. Select **Assign Networks**.
2. From the **Network** dropdown menu, select the network on which you want to run Elastic Runtime.

Network Assignments

Network

default

Save

3. Click **Save**.

Step 3: Configure Domains

1. Select **Domains**.

System Domain *

system.example.com

This domain is for system-level PCF components, such as Apps Manager, service brokers, etc. You must set up a wildcard DNS record for this domain that points to your entry point load balancer or HAProxy.

Apps Domain *

apps.example.com

Save

2. Enter the system and application domains.

- The **System Domain** defines your target when you push apps to Elastic Runtime. For example, `system.example.com`.
- The **Apps Domain** defines where Elastic Runtime 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 Elastic Runtime later on in the installation process. See the

Configuring Resources section.

3. Under **Select one of the following point-of-entry options**, choose the **Forward SSL to HAProxy** option. This sets HAProxy as the point of entry for your environment, behind the external load balancer.
4. Complete the fields required to terminate SSL/TLS at HAProxy and any optional fields. For example, you must provide a**SSL Certificate and Private Key**. See [Configuring SSL/TLS Termination at HAProxy](#) for details.

Select one of the following point-of-entry options:^{*}

Forward SSL to Elastic Runtime Router. Assumes an external load balancer is configured to forward encrypted traffic.

Forward unencrypted traffic to Elastic Runtime Router. Assumes an external load balancer is configured to forward unencrypted traffic.

Forward SSL to HA Proxy. Like first option - Assumes an external load balancer is configured to forward encrypted traffic.

SSL Certificate and Private Key ^{*}

```
-----BEGIN RSA PRIVATE KEY-----
MIIEvQIBAAKCAQEA...[REDACTED]
```

[Generate RSA Certificate](#)

Disable HTTP traffic to HAProxy

HAProxy SSL Ciphers

Request Max Buffer Size ^{*}

Disable SSL certificate verification for this environment

For details on

generating certificates for your wildcard system domains, see the [Providing a Certificate for your SSL Termination Point](#) topic.

5. 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.

6. Select the **Disable insecure cookies on the Router** checkbox to set the secure flag for cookies generated by the router.
7. 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](#).

Disable SSL certificate verification for this environment

Disable insecure cookies on the Router

Enable Zipkin tracing headers on the router

8. 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.

9. The **Loggregator Port** defaults to `443` if left blank. Leave this field blank.
10. (Optional) Use the **Applications Subnet** field if you need to avoid address collision with a third-party service on the same subnet as your apps. Enter a CIDR subnet mask specifying the range of available IP addresses assigned to your app containers. The IP range must be different from the network used by the system VMs.
11. (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.

Choose whether to enable route services. Route services enable you to proxy requests to your app over TLS to arbitrary URLs before hitting your app.*

Enable route services
 Disable route services

Loggregator Port

Applications Subnet (Only change this if you need to avoid address collision with a third-party service on the same subnet.) *

Applications Network Maximum Transmission Unit (MTU) (in bytes) *

12. (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](#).

13. (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.
14. (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.

Router Timeout to Backends (in seconds) (min: 1) *

Load Balancer Unhealthy Threshold *

Load Balancer Healthy Threshold *

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

Router Max Idle Keepalive Connections (min: 0, max: 50000) *

16. TCP Routing is disabled by default. To enable this feature, perform the following steps:
 - a. Select the **Enable TCP Routing** radio button.
 - 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.

Enable TCP requests to your apps via specific ports on the TCP router. You will want to configure a load balancer to forward these TCP requests to the TCP routers. If you do not have a load balancer, then you can also send traffic directly to the TCP router.*

- Select this option if you prefer to enable TCP Routing at a later time
- Enable TCP Routing

TCP Routing Ports (one-time configuration, if you want to update this value you can via the CF CLI) *

`1024-1123`

- 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 Elastic Runtime. See [Configuring Resources](#).

17. To disable TCP routing, select the **Select this option if you prefer to enable TCP Routing at a later time** radio button. For more information, see the [Configuring TCP Routing in Elastic Runtime](#) topic.

18. Click **Save**.

Step 5: Configure Application Containers

1. Select **Application Containers**.

Enable microservice frameworks, private Docker registries, and other services that support your applications at a container level.

- Enable Custom Buildpacks
- Allow SSH access to app containers
- Enable SSH when an app is created

Private Docker Insecure Registry Whitelist

`10.10.10.10:8888,example.com:8888`



Docker Images Disk-Cleanup Scheduling on Cell VMs*

- Never clean up Cell disk-space
- Routinely clean up Cell disk-space
- Clean up disk-space once threshold is reached

Threshold of Disk-Used (MB) (min: 1) *

Max Inflight Container Starts *

Save

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 Elastic Runtime 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 Trusted Registries](#) topic for more information.
6. Select your preference for **Docker Images Disk-Cleanup on Cell VMs**. If you choose **Clean up disk-space once threshold is reached** enter a **Threshold of Disk-Used** in megabytes.
7. Optionally, 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. Click **Save**.

Step 6: Configure Application Developer Controls

1. Select **Application Developer Controls**.

Configure restrictions and default settings for applications pushed to Elastic Runtime.

Maximum File Upload Size (MB) (min: 1024, max: 2048) *	<input type="text" value="1024"/>
Default App Memory (MB) (min: 64, max: 2048) *	<input type="text" value="1024"/>
Default App Memory Quota per Org (min: 10240, max: 102400) *	<input type="text" value="10240"/>
Maximum Disk Quota per App (MB) (min: 512, max: 10240) *	<input type="text" value="2048"/>
Default Disk Quota per App (MB) (min: 512, max: 10240) *	<input type="text" value="1024"/>
Default Service Instances Quota per Org (min: 0, max: 1000) *	<input type="text" value="100"/>
<input type="button" value="Save"/>	

2. Enter your intended maximum file upload size.
3. Enter your default RAM memory allocation per app.
4. Enter your default total RAM memory (RAM) quota per Org. You can change this in the CLI.
5. Enter your maximum and default disk quotas per app.
6. Enter your default service instances quota per Org. You can change this in the CLI.

7. 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 Elastic Runtime deployment completes, you will review and set the appropriate application security groups.

Setting appropriate Application Security Groups that control application network policy is the responsibility of the Elastic Runtime administration team. Please refer to the Application Security Groups topic in the Pivotal Cloud Foundry documentation for more detail on completing this activity after the Elastic Runtime deployment completes.

Type X to acknowledge that you understand this message *

Save

Step 8: Configure Authentication and Enterprise SSO

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

Configure your user store access, which can be an internal user store (managed by Cloud Foundry's UAA) or an external user store (LDAP or SAML). You can also adjust the lifetimes of authentication tokens.

Configure your UAA user account store with either internal or external authentication mechanisms*

- Internal UAA (provided by Elastic Runtime; configure your password policy below)

Minimum Password Length *

[x]

Minimum Uppercase Characters Required for Password *

Minimum Lowercase Characters Required for Password *

Minimum Numerical Digits Required for Password *

Minimum Special Characters Required for Password *

Number of Months Before Password Expires *

Maximum Password Entry Attempts Allowed *

SAML Identity Provider

LDAP Server

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 Elastic Runtime](#) 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 Elastic Runtime](#) topic.

3. (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.

4. (Optional) Customize the text prompts used for username and password from the cf CLI and Apps Manager login popup.

5. (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 public IP address(es), append a regular expression or regular expressions to match the public IP address(es).

Apps Manager Access Token Lifetime (in seconds) *

Apps Manager Refresh Token Lifetime (in seconds) *

Cloud Foundry CLI Access Token Lifetime (in seconds) *

Cloud Foundry CLI Refresh Token Lifetime (in seconds) *

Set the lifetime of the refresh token for the Cloud Foundry CLI.

Customize Username Label (on login page) *

Customize Password Label (on login page) *

Proxy IPs Regular Expression *

Save

6. Click **Save**.

Step 9: Configure System Databases

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

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**.

Place the databases used by Elastic Runtime components like Cloud Controller and UAA.

Choose the location of your system databases*

Internal Databases - MySQL and Postgres (the Postgres DBs are not highly-available, but this selection is required if you want to keep your system data from a pre-1.6.0 Elastic Runtime that you upgraded)

Internal Databases - MySQL (preferred for complete high-availability)

External Databases (preferred if, for example, you use AWS RDS)

Save

2. Select **Internal Databases - MySQL**.

3. Click **Save**.

Then proceed to [Step 10: \(Optional\) Configure Internal MySQL](#) 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. You can configure a different database provider that provides MySQL support, such as Google Cloud SQL.

To create your Elastic Runtime 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 seven Elastic Runtime components that require a relational database:

```
CREATE database uaa;
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;
```

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

6. In Elastic Runtime, select **Databases**.

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

Place the databases used by Elastic Runtime components like Cloud Controller and UAA.

Choose the location of your system databases*

- Internal Databases - MySQL and Postgres (the Postgres DBs are not highly-available, but this selection is required if you want to keep your system data from a pre-1.6.0 Elastic Runtime that you upgraded)
- Internal Databases - MySQL (preferred for complete high-availability)
- External Databases (preferred if, for example, you use AWS RDS)

Hostname *

TCP Port *

App Usage Service database username *

App Usage Service database password *

Secret

8. Complete the following fields in Elastic Runtime:

Elastic Runtime Field	Notes

Hostname	Specify the hostname of the database server.
TCP Port	Specify the port of the database server.
DATABASE-NAME database username	Specify a unique username that can access this specific database on the database server.
DATABASE-NAME database password	Specify a password for the provided username.

9. Click **Save**.

Step 10: (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 [Proxy](#) section of the *MySQL for PCF* topic for more information.

Only configure this section if you selected Internal Databases - MySQL or Internal Databases - MySQL and Postgres in the previous Databases section.

A proxy tier routes MySQL connections from internal components to healthy cluster nodes. Configure DNS and/or your own load balancer to point to multiple proxy instances for increased availability. TCP healthchecks can be configured against port 1936.

The automated backups functionality works with any S3-compatible file store that can receive your backup files.

MySQL Proxy IPs

MySQL Service Hostname

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.
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.

Replication canary time period *
<input type="text" value="30"/>
Replication canary read delay *
<input type="text" value="20"/>
E-mail address (required) *
<input type="text" value="email@example.com"/>

7. Under **Automated Backups Configuration**, choose one of three options for MySQL backups:

- **Disable automatic backups of MySQL**
- **Enable automated backups from MySQL to an S3 bucket or other S3-compatible file stores**saves your backups to an existing Amazon Web Services (AWS) or [Ceph](#) S3-compatible blobstore.

Automated Backups Configuration*
<input type="radio"/> Disable automated backups of MySQL <input checked="" type="radio"/> Enable automated backups from MySQL to an S3 bucket or other S3-compatible file store
S3 Endpoint URL *
<input type="text" value="https://s3.amazonaws.com"/>
S3 Bucket Name *
<input type="text" value="my-bucket"/>
Bucket Path *
<input type="text" value="my-backups-folder/elastic-runtime"/>
AWS Access Key ID *
<input type="text"/>
AWS Secret Access Key *
<input type="text" value="Secret"/>
Cron Schedule *
<input type="text" value="@every 15m"/>
<input checked="" type="checkbox"/> Backup All Nodes
<input type="radio"/> Enable automated backups from MySQL to a remote host via SCP

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 **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 automated backups from MySQL to a remote host via SCP**saves your backups to a remote host using secure copy

Automated Backups Configuration*

Disable automated backups of MySQL

Enable automated backups from MySQL to an S3 bucket or other S3-compatible file store

Enable automated backups from MySQL to a remote host via SCP

Hostname *

Port *

Username *

Private key *

Destination directory *

Cron Schedule *

Backup All Nodes

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 Elastic Runtime tile to 0.

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

- 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.

Server Activity Logging*

Disable server activity logging
 Enable server activity logging

Event types *

connect,query

Load Balancer Healthy Threshold *

0

Load Balancer Unhealthy Threshold *

0

Save

9. 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.

10. Click **Save**.

Step 11: Configure File Storage

To minimize system downtime, Pivotal recommends using highly resilient and redundant *external* filestores for your Elastic Runtime file storage.

When configuring file storage for the Cloud Controller in Elastic Runtime, 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 Elastic Runtime tile, select **File Storage**.
2. Select **Internal WebDAV**, and click **Save**.

External Azure Storage

To use external Azure file storage for your Elastic Runtime filestore, perform the following steps:

1. Select the **External AzureStorage** option.

Configure your Cloud Controller's filesystem*

- Internal WebDAV (provided by Elastic Runtime)
- External S3-Compatible File Store (if you want to use a service like S3 or Ceph)
- External Google Cloud Storage
- External Azure Storage

Account Name *

Access Key *

Buildpacks Container Name *

Droplets Container Name *

Packages Container Name *

Resources Container Name *

2. To create a new storage account and storage containers for the Elastic Runtime 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 Elastic Runtime.
- 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 Elastic Runtime, 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 12: (Optional) Configure System Logging

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

1. Select **System Logging**.

Configure system logging. Complete the External Syslog fields only if using an external syslog server.

Enable Cloud Controller security event logging

External Syslog Aggregator Hostname
 514

External Syslog Aggregator Port
 The typical syslog port is 514. Ensure syslogd is listening on external interfaces.

External Syslog Network Protocol

Syslog Drain Buffer Size (# of messages) *

Save

2. 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).
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 Elastic Runtime 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**.

Step 13: (Optional) Customize Apps Manager

The **Custom Branding** and **Apps Manager** sections customize the appearance and functionality of Apps Manager. Refer to [Custom Branding](#) and [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.

Customize colors, images, and text for Apps Manager and the Cloud Foundry login portal.

Company Name

Accent Color

Main Logo (PNGs only)

Square Logo/Favicon (PNGs only)

Footer Text
 Defaults to 'Pivotal Software Inc. All rights reserved.'

Footer Links
You may configure up to three links in the Apps Manager footer Add

Classification Header/Footer Background Color

Classification Header/Footer Text Color

Classification Header Content

Classification Footer Content

Save

2. Click **Save** to save your settings in this section.
3. Select **Apps Manager**. Use this section to configure page names and sidebar links in the **Apps Manager** and **Marketplace** pages.

Configure Apps Manager

Display Marketplace Service Plan Prices

Supported currencies as json *****

Define the currency codes and associated symbols (defaults to "{ \"usd\": \"\$\", \"eur\": \"€\" }" if blank)

Product Name

Marketplace Name

Customize Sidebar Links

You may configure up to 10 links in the Apps Manager sidebar

Add

▶ Marketplace	
▶ Docs	
▶ Tools	

Save

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

Step 14: (Optional) Configure Email Notifications

Elastic Runtime 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**.

Configure Simple Mail Transfer Protocol for the Notifications application to send email notifications about your deployment. This application is deployed as an errand in Elastic Runtime. If you do not need this service, leave this section blank and disable the Notifications and Notifications UI errands.

From Email

Address of SMTP Server

Port of SMTP Server

SMTP Server Credentials

[Change](#)

SMTP Enable Automatic STARTTLS

SMTP Authentication Mechanism*

SMTP CRAMMD5 secret

Save

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 15: (Optional) Add CCDB Restore Key

Perform this step if all of the following are true:

- You deployed Elastic Runtime previously.
- You then stopped Elastic Runtime or it crashed.
- You are re-deploying Elastic Runtime with a backup of your Cloud Controller database.

1. Click **Restore CCDB Encryption Key**.
2. Enter your **Cloud Controller DB Encryption Key**.

You must enter your original database encryption key here once you have deployed Elastic Runtime for the first time. Anytime thereafter your encryption key will remain the same, if you specify the same key here. This is important for backup and restore of data as well as upgrades.

Cloud Controller DB Encryption Key

...

Leave blank unless restoring a Cloud Controller Database exported from a previous installation.

Save

See [Backing Up Pivotal Cloud Foundry](#) for more information.

Step 16: Configure Smoke Tests

The Smoke Tests errand runs basic functionality tests against your Elastic Runtime 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 org and space**, which creates an ad-hoc org and space for running smoke tests and deletes them afterwards. Otherwise, select **Specified org and space** and complete the fields to specify where you want to run smoke tests.

Specify a Cloud Foundry organization and space where smoke tests can run if in the future you delete your Elastic Runtime deployment domains.

Choose whether to create a new org and space on demand for smoke tests*

- Temporary org and space (This org and space are deleted after smoke tests finish.)
 Specified org and space (The org and space must have a domain available for routing.)

Organization *

Space *

Domain *

Save

3. Click **Save**.

Step 17: (Optional) Enable Advanced Features

The Advanced Features section of Elastic Runtime 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

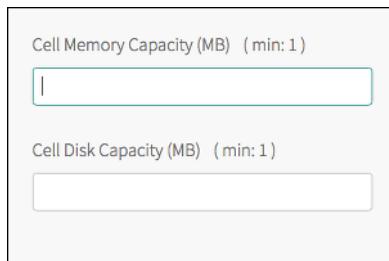
If your apps do not use the full allocation of disk space and memory set in the **Resource Config** tab, you might want use this feature. These fields control the amount to overcommit disk and memory resources to each Diego Cell VM.

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**.



The screenshot shows a form with two input fields. The first field is labeled "Cell Memory Capacity (MB) (min: 1)" with a text input box containing a single vertical bar character. The second field is labeled "Cell Disk Capacity (MB) (min: 1)" with an empty text input box.

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

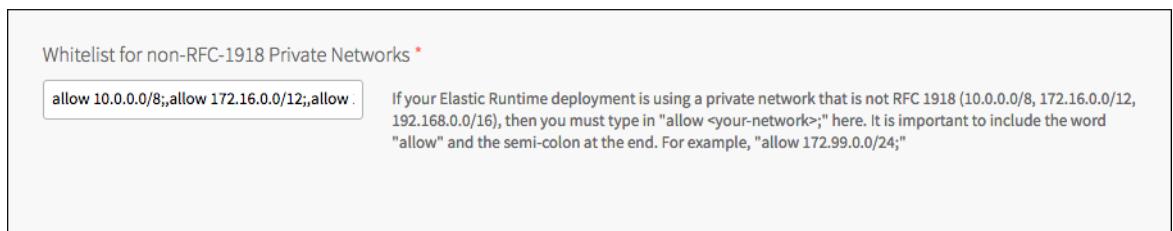
Some private networks require extra configuration so that internal file storage (WebDAV) can communicate with other PCF processes.

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.



The screenshot shows a text input field for the whitelist. The placeholder text is "Whitelist for non-RFC-1918 Private Networks *". Below the input field is a note: "allow 10.0.0.0/8;allow 172.16.0.0/12;allow . If your Elastic Runtime deployment is using a private network that is not RFC 1918 (10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16), then you must type in "allow <your-network>;" here. It is important to include the word "allow" and the semi-colon at the end. For example, "allow 172.99.0.0/24;"

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 Elastic Runtime 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 Elastic Runtime.

To modify the value of the **CF CLI Connection Timeout**, perform the following steps:

1. Select **Advanced Features**.

A screenshot of a configuration dialog titled "CF CLI Connection Timeout". It contains a single input field with the value "15" entered.

2. Add a value, in seconds, to the **CF CLI Connection Timeout** field.
3. Click **Save**.

Container-to-Container Networking

Enabling Container-to-Container Networking allows direct, policy-driven communication between containers on an overlay network occupying a specific range. For more information about Container-to-Container Networking, see the [Administering Cloud Foundry Networking](#) topic.

By default, PCF sets the overlay range to `10.255.0.0/16`, but you can modify this range when you enable Container to Container Networking.

Enable Container-to-Container Networking

To enable Container-to-Container Networking, perform the following steps:

1. Select **Advanced Features**.
2. Select **Enable Container-to-Container Networking**. Optionally, add a range of IP addresses in CIDR format to use instead of the default

A screenshot of a configuration dialog for enabling Container-to-Container Networking. It includes a note about enabling the network, a radio button for "Enable Container-to-Container Networking" (which is selected), and a text input field for "Network CIDR" containing the value "10.255.0.0/16".

overlay network.

3. Click **Save**.

Disable Container-to-Container Networking

To disable Container-to-Container Networking, perform the following steps:

1. Select **Advanced Features**.
2. Select **Disable Container-to-Container Networking**.

A screenshot of a configuration dialog for disabling Container-to-Container Networking. It includes a note about enabling the network, two radio buttons ("Enable Container-to-Container Networking" and "Disable Container-to-Container Networking", with "Disable" selected), and a "Save" button at the bottom.

3. Click **Save**.

CF API Rate Limiting

Enabling 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.

Enable CF API Rate Limiting

To enable CF API Rate Limiting, perform the following steps:

1. Select **Advanced Features**.
2. Under **Enable CF API Rate Limiting**, select **Enable**.

Enabling CF API Rate Limiting will prevent 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.*

Enable

General Limit *

2000

Unauthenticated Limit *

100

3. 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**.
4. For **Unauthenticated Limit**, enter the number of requests an unauthenticated client is allowed to make over an hour interval. The default value is **100**.
5. Click **Save**.

Disable CF API Rate Limiting

To disable CF API Rate Limiting, perform the following steps:

1. Select **Advanced Features**.
2. Under **Enable CF API Rate Limiting**, select **Disable**.

Enabling CF API Rate Limiting will prevent 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.*

Enable

Disable

3. Click **Save**.

Step 18: Configure Errands

Errands are scripts that Ops Manager runs automatically when it installs or uninstalls a product, such as a new version of Elastic Runtime. There are two types of errands: *post-deploy errands* run after the product is installed, and *pre-delete errands* run before the product is 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. The Elastic Runtime 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.

Errands

Errands are scripts that run at designated points during an installation.

Post-Deploy Errands

Smoke Test Errand

Runs Smoke Tests against your Elastic Runtime installation

On

Apps Manager Errand

Pushes the Pivotal Apps Manager application to your Elastic Runtime installation

On

Notifications Errand

Pushes the Pivotal Notifications application to your Elastic Runtime installation

On

Notifications UI Errand

Pushes the Notifications UI component to your Elastic Runtime installation

On

Pivotal Account Errand

Pushes the Pivotal Account application to your Elastic Runtime installation

On

Autoscaling Errand

Pushes the Pivotal App Autoscaling application to your Elastic Runtime installation

On

Autoscaling Registration Errand

Registers the Autoscaling Service Broker

On

NFS Broker Errand

Pushes the NFS Broker application to your Elastic Runtime installation

On

There are no pre-delete errands for this product.

Save

- **Smoke Test Errand** verifies that your deployment can do the following:
 - Push, scale, and delete apps
 - Create and delete orgs and spaces
- **Apps Manager Errand** deploys the Apps Manager, a dashboard for managing apps, services, orgs, users, and spaces. Until you deploy Apps Manager, you must perform these functions through the of CLI. After Apps Manager has been deployed, Pivotal recommends deselecting the checkbox for this errand on subsequent Elastic Runtime deployments. For more information about the 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 Elastic Runtime. See the [Enabling NFS Volume Services](#) topic for more information.

Step 19: Configure Resources

1. Select **Resource Config**.

Resource Config				
JOB	INSTANCES	PERSISTENT DISK TYPE	VM TYPE	LOAD BALANCER
Consul	Automatic: 1	Automatic: 1 GB	Automatic: Standard_F1s (cpu: 1, ram: 2 GB, disk: 1)	
NATS	Automatic: 1	None	Automatic: Standard_F1s (cpu: 1, ram: 2 GB, disk: 1)	
etcd	Automatic: 1	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: 1	None	Automatic: Standard_F1s (cpu: 1, ram: 2 GB, disk: 1)	
MySQL Server	Automatic: 1	Automatic: 100 GB	Automatic: Standard_DS1_v2 (cpu: 2, ram: 14 GB, disk: 1)	
Backup Prepare Node	0	Automatic: 200 GB	Automatic: Standard_F1s (cpu: 1, ram: 2 GB, disk: 1)	
Apps Manager Database (Postgres)	Automatic: 0	Automatic: 1 GB	Automatic: Standard_F1s (cpu: 1, ram: 2 GB, disk: 1)	
Cloud Controller Database (Postgres)	Automatic: 0	Automatic: 2 GB	Automatic: Standard_F1s (cpu: 1, ram: 2 GB, disk: 1)	
UAA Database (Postgres)	Automatic: 0	Automatic: 10 GB	Automatic: Standard_F1s (cpu: 1, ram: 2 GB, disk: 1)	
UAA	Automatic: 1	None	Automatic: Standard_F2s (cpu: 2, ram: 4 GB, disk: 3)	
Cloud Controller	Automatic: 1	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: 1	None	Automatic: Standard_F1s (cpu: 1, ram: 2 GB, disk: 1)	

2. Retrieve the name(s) of your load balancer by navigating to the Azure portal, clicking **All resources**, and locating your **Load balancer** resource.

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.

3. Locate the **HAProxy** job in the **Resource Config** pane and enter the name of your load balancer in the field under the **Load Balancer** column. If you followed the [procedure](#) in the previous topics, the name of the load balancer should be `pcf-lb`.

HAProxy	Automatic: 1	None	Automatic: Standard_F1s (cpu: 1, ram: 2 G)	pcf-lb
Router	Automatic: 1	None	Automatic: Standard_F1s (cpu: 1, ram: 2 G)	

Note: Do not enter a load balancer for the **Diego Brain** component. HAProxy routes SSH traffic to apps.

4. Ensure that the **Internet Connected** checkboxes are selected for all jobs. This gives all VMs a public IP address that enables outbound Internet access.

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. Azure offers a managed [source NAT](#).

(SNAT) service [\[x\]](#). However, Pivotal does not recommend using this service because Elastic Runtime deployments may fail due to SNAT performance bottlenecks.

5. 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 Elastic Runtime 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 20: (Optional) Disable Unused Resources

By default, Elastic Runtime uses an internal filestore and internal databases. If you configure Elastic Runtime to use external resources, you can disable the corresponding system-provided resources in Ops Manager to reduce costs and administrative overhead.

For more information regarding scaling instances, see the [Zero Downtime Deployment and Scaling in CF](#) and the [Scaling Instances in Elastic Runtime](#) topics.

Complete the following procedures to disable specific VMs in Ops Manager:

1. Click **Resource Config**.
2. If you configure Elastic Runtime to use an external S3-compatible filestore, edit the following fields:
 - **File Storage:** Enter in **Instances**.
3. If you configure Elastic Runtime to use an external Relational Database Service (RDS), edit the following fields:
 - **MySQL Proxy:** Enter in **Instances**.
 - **MySQL Server:** Enter in **Instances**.
 - **Cloud Controller Database:** Enter in **Instances**.
 - **UAA Database:** Enter in **Instances**.
4. Click **Save**.

Step 21: Configure Stemcell

Verify whether Ops Manager is providing the stemcell version required by Elastic Runtime. If the correct version is already present, you do not need to download a new stemcell.

1. In the Elastic Runtime tile, select **Stemcell**.
2. Verify that the version indicated in the filename matches the version of stemcell required by Elastic Runtime.
 - If Elastic Runtime detects that a stemcell `.tgz` file is present in the Ops Manager Director VM at `/var/tempest/stemcells/`, the Stemcell screen displays filename information.

Stemcell

A stemcell is a template from which Ops Manager creates the VMs needed for a wide variety of components and products.

cf requires BOSH stemcell version 3262 ubuntu-trusty

✓ Using `bosh-stemcell-3262.4-vsphere-esxi-ubuntu-trusty-go_agent.tgz`

Import Stemcell

- If Elastic Runtime cannot detect a stemcell `.tgz` file, the following message displays:

Stemcell

A stemcell is a template from which Ops Manager creates the VMs needed for a wide variety of components and products.

cf requires BOSH stemcell version 3262 ubuntu-trusty

[✖ Go to Pivotal Network and download Stemcell 3262.12 ubuntu-trusty.](#)

[Import Stemcell](#)

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 Elastic Runtime, 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 Elastic Runtime tile, click **Import Stemcell** to import the downloaded stemcell `.tgz` file.

Step 22: Complete the Elastic Runtime 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 screenshot shows the PCF Ops Manager dashboard. At the top, there's a navigation bar with a teal 'P' logo and the text 'PCF Ops Manager'. On the right side of the bar, it says 'pivotal-cf'. Below the bar, a red alert box contains the text: '⚠ Please review the errors below' followed by a bulleted list: '• Cannot reach gateway with IP 10.0.0.16.1 (ignorable if ICMP is disabled)' and '• Cannot reach DNS with IP 10.0.0.2 (ignorable if ICMP is disabled)'. To the right of the list are two buttons: 'Ignore errors and start the install' (in white) and 'Stop and fix errors' (in red).

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.

The screenshot shows a modal window titled 'Changes Applied' with a green checkmark icon. The main message inside the window reads: 'Ops Manager Director was successfully installed. We recommend that you export a backup of this installation from the actions menu.' At the bottom of the window are two buttons: 'Close' (gray) and 'Return to Installation Dashboard' (blue). The background of the window shows a progress bar at 100% completion and a list of installation steps: 'Setting Micro BOSH', 'Installing', 'Checking Micro BOSH status', and 'Logging into director'. The overall background of the page is dark teal.

Deploying PCF on Azure Government Cloud

Page last updated:

This topic describes how to install Pivotal Cloud Foundry (PCF) on Azure Government Cloud.

 **Note:** Azure Government Cloud is only supported in PCF 1.10 and later.

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: Prepare to Deploy

Perform the procedures in the [Preparing to Deploy PCF on Azure](#), but when using `azure login`, always specify the `AzureUSGovernment` environment:

```
$ azure login --environment AzureUSGovernment
```

Step 2: Launch an Ops Manager Director Instance

Perform the procedures in the [Launching an Ops Manager Director Instance on Azure without an ARM Template](#) topic.

Step 3: 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 4: Configure Ops Manager](#).

External Identity Provider

If you configured your Ops Manager for an external Identity Provider with SAML, perform the following steps:

- From your local machine, target your Ops Manager UAA server:

```
$ uaac target https://OPS-MAN-FQDN/uaa
```

- 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.`

- Continue to [Step 4: Configure Ops Manager](#).

Step 4: Configure Ops Manager

Perform the following steps to configure Ops Manager for Azure Government Cloud:

- List your tokens.

```
$ uaac contexts
```

Locate the entry for your Ops Manager FQDN. Under `client_id: opsman`, record the value for `access_token`.

- 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 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.

- If the `curl` command returns a `200 OK` response, navigate to the Ops Manager FQDN in a browser and log in.
- 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.
- 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
    }
  ]
}
```
- 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
```

- 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
    }
  ]
}
```

- Return to the Ops Manager Installation Dashboard by navigating to the Ops Manager FQDN in a browser.
- Click **Apply Changes** to redeploy Ops Manager.
- When Ops Manager finishes deploying, continue to the [Deploying Elastic Runtime on Azure](#) topic to deploy Elastic Runtime and 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

Because HAProxy sits between the ALB and the router, you can resolve the problem by setting the HAProxy timeout value to be greater than the timeout value of the ALB. Perform the following steps:

1. Target your BOSH Director by following the steps in the [Log into BOSH](#) section of the *Advanced Troubleshooting with the BOSH CLI* topic.
2. If necessary, download your PCF manifest:

```
$ bosh download manifest DEPLOYMENT-NAME
```

If you do not know your deployment name, run `bosh deployments` to display a list of available deployments.

3. Set your current deployment to your PCF manifest:

```
$ bosh deployment PATH-TO-PCF-MANIFEST
```

4. Edit your deployment:

```
$ bosh edit deployment
```

5. Locate the `haproxy` section and add the following property: `request_timeout_in_seconds: 180`

6. Save and exit the edited manifest.

7. Redeploy:

```
$ bosh deploy
```

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 0.9.18 or greater. Run `az --version` from the command line to display your current Azure CLI version.

Solution

Upgrade your Azure CLI to the current version by reinstalling the new version. Run `npm update -g azure-cli` from the command line, or follow the procedures below for your operating system:

- **Mac OS X:** Download and run the [Mac OS X](#) installer.
- **Windows:** Download and run the [Windows](#) installer. Install the Azure CLI on Windows 10. Use the command line, not the PowerShell, to run the Azure CLI.
- **Linux:** Download the [Linux](#) tar file. Install the Azure CLI on Ubuntu Server 14.04 LTS. To install the Azure CLI on Linux, you must first install Node.js and npm, and then run `sudo npm install -g PATH-TO-TAR-FILE`.

Deployment Fails at “bosh-init”

Symptom

After clicking **Apply Changes** to install Ops Manager and Elastic Runtime, the deployment fails at `bosh-init` 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 set:false omitEmpty:false} pkcs1PrivateKey @2  
===== 2016-09-29 16:28:22 UTC Finished "bosh-init deploy  
/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 a PCF on Azure Installation

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](#).

Installing Pivotal Cloud Foundry on GCP

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](#).

Prerequisites

The following sections describe general requirements for running PCF and specific requirements for running PCF on GCP.

General Requirements

The following are general requirements for deploying and managing a PCF deployment with Ops Manager and Elastic Runtime:

- (**Recommended**) Ability to create a wildcard DNS record to point to your router or load balancer. Alternatively, you can use a service such as xip.io. For example, `203.0.113.0.xip.io`.
Elastic Runtime 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 router, every application deployed to the `example.com` domain resolves to the IP address of your router.
- (**Recommended**) A network without DHCP available for deploying the Elastic Runtime VMs

 **Note:** If you have DHCP, refer to the Troubleshooting Guide to avoid issues with your installation.

- Sufficient IP allocation:
 - One IP address for each VM instance
 - An additional IP address for each instance that requires static IPs
 - An additional IP address for each errand
 - An additional IP address for each compilation worker: `IPs needed = VM instances + static IPs + errands + compilation workers`
-  **Note:** BOSH requires that you allocate a sufficient number of additional dynamic IP addresses when configuring a reserved IP range during installation. BOSH uses these IPs during installation to compile and deploy VMs, install Elastic Runtime, and connect to services. We recommend that you allocate at least 36 dynamic IP addresses when deploying Ops Manager and Elastic Runtime.
- The most recent version of the [Cloud Foundry Command Line Interface \(cf CLI\)](#)
- One or more NTP servers if not already provided by your IaaS

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 ERT.

- 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. Installing PCF requires at least 10 concurrent instances for testing or development deployments and typically at least 30 for production deployments. The exact number of instances depends on the number of tiles and availability zones you plan to deploy. A minimal single-AZ GCP deployment requires the following 20 [custom](#) VMs:

VM Count	Machine type	Memory (in GB)
11	1vCPU	1.00
1	1vCPU	6.00
2	1vCPU	2.00
2	2vCPU	4.00
1	2vCPU	8.00
3	4vCPU	16.00

- 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.

Install PCF on GCP

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. [\(Optional\) Installing the PCF IPsec Add-On](#)
5. [Deploying Elastic Runtime on GCP](#)

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 will not have enough quota for a typical production-level PCF deployment.

To view production-level deployment options for PCF on GCP, see the [Reference Architecture for Pivotal Cloud Foundry on GCP](#).

The following table lists recommended resource quotas for a PCF deployment running about 100 app instances:

Resource	Suggested Minimum Quota
CPUs <i>Region Specific</i>	150
Firewall Rules	15
Forwarding Rules	5
Global Forwarding Rules	5
Global Backend Services	5
Health Checks	10
Images	10
Static IP Addresses <i>Region Specific & Assuming SNAT topology</i>	5
IP Addresses Global	5
IP Addresses <i>Region Specific & Assuming SNAT topology</i>	5
Networks	5
Subnetworks	5
Routes	20
Target Pools	10
Target HTTP Proxies Global	5
Target HTTPS Proxies Global	5
Total persistent disk reserved (GB) <i>Region Specific</i>	15,000

For instructions on how to set up GCP resources required to deploy PCF, see [Preparing to Deploy PCF on GCP](#).

Preparing to Deploy PCF on 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. You may also find it helpful to review different deployment options in the [Reference Architecture for Pivotal Cloud Foundry on GCP](#).

Step 1: Create a GCP Network with Subnet

1. Log in to the GCP Console at <https://console.cloud.google.com>.
2. In the console, navigate to the GCP project where you want to install PCF.
3. Select **Networking**, then **Create Network**.
 - a. Enter a name and description for your network.
 - b. Under **Subnetworks**, select **Custom** and specify a name, region, and address range in CIDR notation.

Make sure you select a region with enough zones to support the availability zone needs of your deployment. For help selecting the correct region for your deployment, see the [Google documentation on regions and zones](#).

The screenshot shows the 'Create a network' dialog in the GCP Console. On the left is a sidebar with icons for Compute Engine, Cloud Storage, Cloud Functions, Cloud Pub/Sub, and Cloud Dataflow. The main area has a title 'Create a network' with a back arrow. It contains fields for 'Name' (opsmgr), 'Description (Optional)', and a 'Subnetworks' section. The 'Subnetworks' section includes a 'Custom' tab selected, a 'Name' field (opsmgr-subnet), an 'Add a description' link, a 'Region' dropdown (us-central1), and an 'IP address range' input (10.0.0.0/20). At the bottom are 'Create' and 'Cancel' buttons, and a note about equivalent REST or command line.

Note: To use the [Single Sign-On for PCF](#) service, you must configure a network with only one subnet.

4. Click **Create**.

Step 2: 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**, then **Create Firewall Rule**.
2. Create a firewall rule to allow all traffic within the subnetwork.
 - **Name:** Enter a name, such as `all-internal`.
 - **Network:** Select the network you created in the section above, [Create a GCP Network with Subnet](#).
 - **Source filter:** Choose **Subnetworks**, then select the subnetwork or subnetworks you defined in the section above.
 - **Allowed protocols and ports:** Enter `tcp:0-65535;udp:0-65535;icmp`.
 - **Target tags:** Not used for this firewall rule. This rule uses subnetwork CIDR rules instead to accommodate on-demand service brokers. These brokers deploy VMs outside of Ops Manager and do not apply VM tags.
3. Create a firewall rule to allow `tcp:22;tcp:80;tcp:443` traffic from any source to VMs tagged with `pcf-opsmanager`.
 - **Name:** Enter `pcf-opsmanager`.
 - **Network:** Select the network you created in the section above, [Create a GCP Network with Subnet](#).
 - **Source filter:** Choose `Allow from any source (0.0.0.0/0)`.
 - **Allowed protocols and ports:** Enter `tcp:22;tcp:80;tcp:443`.
 - **Target tags:** Enter `pcf-opsmanager`.
4. Create a firewall rule to allow `tcp:80;tcp:443;tcp:2222;tcp:8080` traffic from any source to VMs tagged with `pcf-lb`.
 - **Name:** Enter `pcf-lb`.
 - **Network:** Select the network you created in the section above, [Create a GCP Network with Subnet](#).
 - **Source filter:** Choose `Allow from any source (0.0.0.0/0)`.
 - **Allowed protocols and ports:** Enter `tcp:80;tcp:443;tcp:2222;tcp:8080`.
 - **Target tags:** Enter `pcf-lb`.
5. If you plan to enable the TCP routing feature, create another firewall rule to allow incoming TCP traffic to the TCP router.
 - **Name:** Enter a name, such as `pcf-tcp-lb`.
 - **Network:** Select the network you created in the section above, [Create a GCP Network with Subnet](#).
 - **Source filter:** Choose `Allow from any source (0.0.0.0/0)`.
 - **Allowed protocols and ports:** Enter `tcp:1024-65535`.
 - **Target tags:** Enter `pcf-cf-tcp`.

Step 3: Set up an IAM Service Account

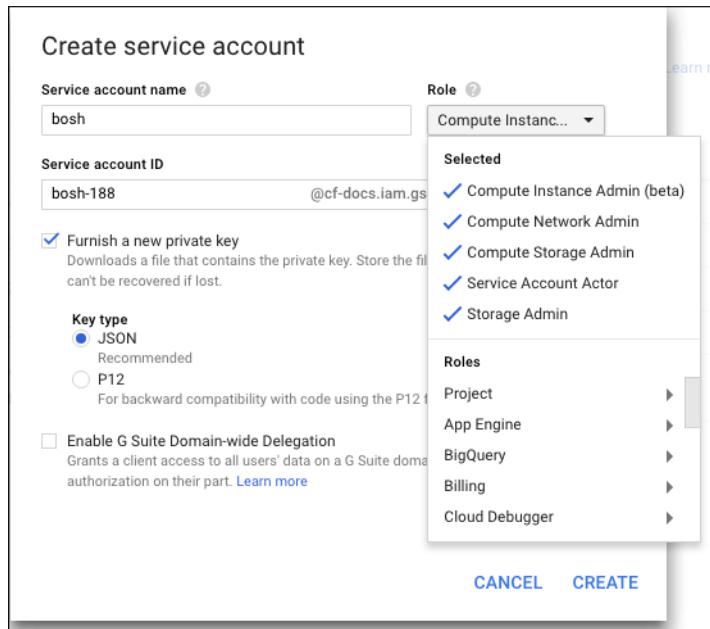
1. From the GCP Console, select **IAM & Admin**, then **Service accounts**.
2. Click **Create Service Account**.
 - **Service account name:** Enter a name. For example, `bosh`.
 - **Role:** Select the following roles for the service account:

 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 autogenerated 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 4: Create a Project-Wide SSH Keypair for Your Project

1. Create an SSH keypair on your local machine with the username `vcap`. For example, use the following command:

```
$ ssh-keygen -t rsa -f vcap-key -C vcap@local
```

When prompted, press enter twice to use no passphrase.

2. Open and copy the contents of the public key file `vcap-key.pub`.
3. In the GCP console, navigate to **Compute Engine > Metadata > SSH Keys**. Click **Add SSH Keys**, or **Edit** if you already have project-wide keys.
4. Paste in the contents of the `vcap-key.pub` file. The username `vcap` autopopulates the username field.
5. Click **Save**.
6. Verify that the public key data is uploaded to your project:
 - a. If you have not done so already, install and set up [gcloud compute](#).
 - b. Execute the following command:

```
$ gcloud compute project-info describe
```

The command outputs project metadata with the new key data appearing as a value in the `sshKeys` field.

```
commonInstanceMetadata:
fingerprint: #####
items:
- key: sshKeys
value: |
  vcap:ssh-rsa ...
```

Step 5: 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 6: Create Load Balancers in GCP

You need at least three and as many as four load balancers to operate PCF on GCP, as follows:

- [HTTP\(S\) Load Balancer](#): For HTTP(S) load balancing, using TCP port `80` and `8080`.
- [SSH Load Balancer](#): For SSH proxy jobs, using TCP port `2222`.
- [TCP WebSockets Load Balancer](#): For WebSockets log tailing, using TCP port `443`.
- [TCP Router](#): Optionally, for apps that use the TCP routing feature.

The steps required to set up each load balancer are described below.

Create Instance Groups and the HTTP(S) Load Balancer

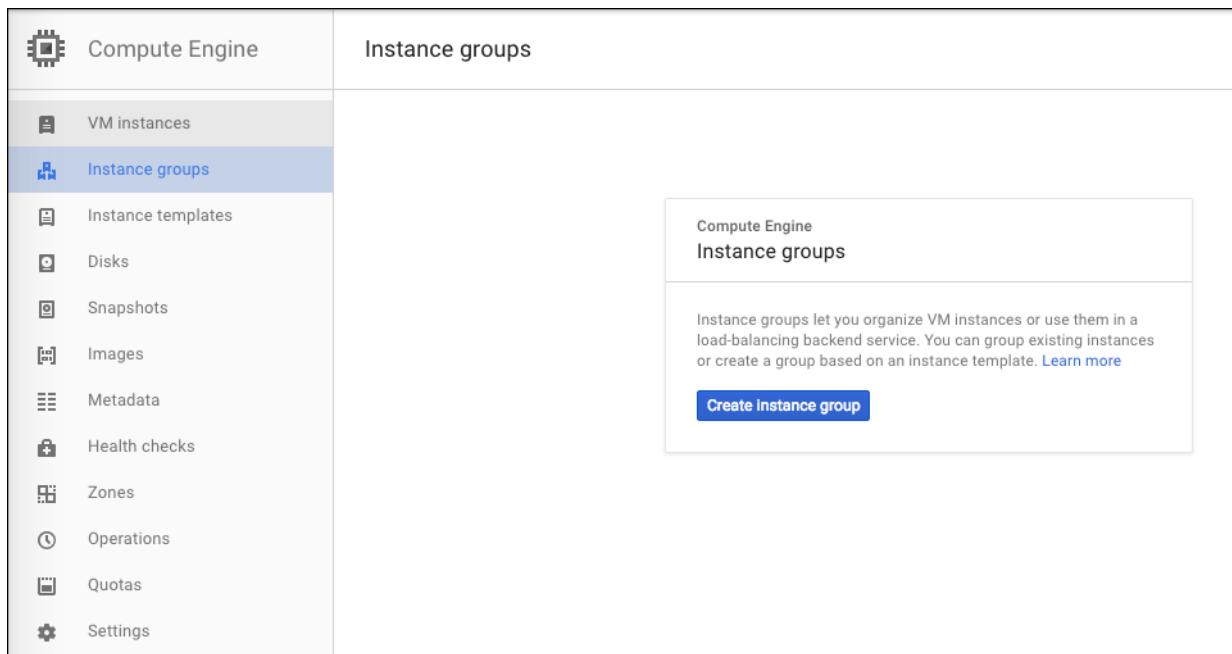
To configure HTTP(S) load balancing for PCF on GCP you need to follow two steps:

1. Create one or more [Instance Group\(s\)](#) for load balancer configuration to the GCP **Backend service**.
2. Create an [HTTP\(S\) Load Balancer](#).

Create **Instance Group(s)**

You need to create and associate one or more **Instance Group(s)** with the HTTP(S) load balancer you create.

1. From the GCP Console, select **Compute Engine** and click **Instance groups**.



2. Click **Create instance group**.
3. In the **Create a new instance group** window, name the instance group in the **Name** field. If you are creating multiple instance groups, make sure each instance group name has a unique name. For example, you might create the following instance groups:
 - pcf-instance-group-lb-1a
 - pcf-instance-group-lb-1b
 - pcf-instance-group-lb-1c

Note: You need one Google **Instance Group** for each Availability Zone you plan to support. All **Instance Groups** must connect to the Google **Backend Service** to configure your load balancer, described below. For a high availability production installation of PCF, Pivotal recommends using three availability zones.
4. For each individual instance group, choose **Single-zone** in the **Location** section.
5. From the **Zone** drop-down menu, select a zone that matches the **Region** of the **network** you created above. Pick a unique zone for each instance group that you create. For example, if you created the network in the `us-central1` region, you could pick the following zones for your instance groups:
 - pcf-instance-group-lb-1a : us-central1-a
 - pcf-instance-group-lb-1b : us-central1-b
 - pcf-instance-group-lb-1c : us-central1-c
6. Under **Creation method**, choose **Select existing instance** and select `opsmgr` from the **Network** drop-down. You previously created the `opsmgr` network in the **Create a GCP Network with Subnet** step above.

Create a new instance group

Use an instance group when configuring a load-balancing backend service or to group VM instances. [Learn more](#)

Name [?](#)

Description (Optional)

Location
Multi-zone groups span multiple zones which assures higher availability. [Learn more](#)

Single-zone
 Multi-zone

Zone [?](#) [▼](#)

Specify port name mapping (Optional)

Creation method
Use a template to create a group of identical instances that can scale automatically. If you do not use a template, you must add and manage each member yourself. [Learn more](#)

Use instance template
 Select existing instances

Network [?](#) [▼](#)

Subnetwork [?](#) [▼](#)

VM instances [▼](#)

Create **Cancel**

Equivalent [REST](#) or [command line](#)

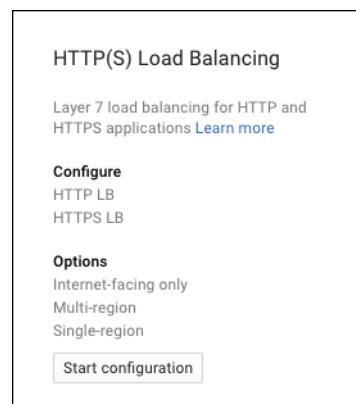
Note: If is your only network, the **Network** drop-down does not appear because the sole network is automatically selected.

7. Click **Create**.
8. If you are creating multiple instance groups, repeat **substeps 2-7** of this procedure.

Create the HTTP(S) Load Balancer

To create a load balancer for HTTP(S) in GCP:

1. From the GCP Console, select **Networking > Load Balancing > Create load balancer**



2. Under **HTTP(S) Load Balancing**, click **Start configuration**.

3. In the **New HTTP(S) load balancer** window, enter `pcf-router` in the **Name** field.

	New HTTP(S) load balancer	Backend configuration
	Name ? <input type="text" value="pcf-router"/>	<p>Create or select a backend service to direct incoming traffic. You can add multiple backend services to serve different types of content.</p> <p>Create or select a backend service </p>
	<ul style="list-style-type: none"> <input checked="" type="radio"/> Backend configuration You have not configured your backend yet <input checked="" type="radio"/> Host and path rules Requests will be sent to the default backend <input checked="" type="radio"/> Frontend configuration Your frontend is configured <input type="radio"/> Review and finalize Optional 	
	Create Cancel	

4. Click **Backend configuration** to configure the **Backend service**.
5. In the **Create or select a backend service** drop-down menu, choose **Create a backend service** to open the **Backend service** window.
6. Fill in the name for your **Backend service** in the **Name** field. Leave **Protocol** set to **HTTP**.

Backend service
Name ? <input type="text" value="pcf-gcp-backend"/>
Description (Optional) <input type="text"/>
Protocol: HTTP Named port: http Timeout: 30 seconds

7. In the **Backends** section, from the **Instance Group** drop-down menu, choose one of the **Instance Group(s)** you created above, and select it.

8. Add port `80` to the **Port numbers** field for PCF to make API calls.

9. If you have created multiple instance groups to support a multiple availability zone PCF deployment, perform the following steps:

- Click the **Add backend** button.
- Select another instance group from the **Instance Group** drop-down menu.
- Specify port `80` again if necessary.
- Repeat until you have selected all the instance groups (three for three availability zones) that you created.

10. From the **Health check** drop-down menu, click to **Create a Health Check** with the following field values:

- **Name:** Enter a name, for example `health-check`, or `pcf-public`.
- **Description:**
- **Protocol:** `HTTP`
- **Port:** `8080`
- **Request path:** `/health`
- Use the default **Health criteria** field values:
 - **Check interval:** `5` seconds
 - **Timeout:** `5` seconds
 - **Healthy threshold:** `2` consecutive successes

Create a health check

Autohealing instance groups and load balancing use health checks to detect when an instance is unresponsive. [Learn more](#)

Name

Description (Optional)

Protocol

Port

Request path

[More](#)

Health criteria

Define how health is determined: how often to check, how long to wait for a response, and how many successful or failed attempts are decisive.

Check interval <input type="text" value="5"/> seconds	Timeout <input type="text" value="5"/> seconds
Healthy threshold <input type="text" value="2"/> consecutive successes	Unhealthy threshold <input type="text" value="2"/> consecutive failures

Save and continue **Cancel**

- **Unhealthy threshold:** consecutive failures

- Click **Save and continue**. The **Backend configuration** section shows a green check mark.

11. Click **Host and path rules** to populate the default fields and a green check mark.

12. Select **Frontend configuration**, and add the following:

- **Protocol:**
- **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, .
 3. Under **IP**, make sure this new static IP address is selected.
 4. Click **Reserve**.
- **Port:**

13. If you are using a trusted SSL certificate or already have a self-signed certificate, proceed to step 15.

14. If you want to use a self-signed certificate generated during [Elastic Runtime network configuration](#), skip over the next step of adding the HTTPS frontend configuration until after you generate the certificate in Elastic Runtime. After you generate the certificate, return to step 15 using the following guidelines:

- Copy and paste the generated contents of the **Router SSL Termination Certificate and Private Key** fields from Elastic Runtime 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.

15. Click **Add frontend IP and port** and add the following:

- **Protocol:**
- **IP:** Select the static IP address you just created for the previous rule.
- **Port:** Leave selected.
- **Certificate:** Select **Create a new certificate**. In the next dialog, perform the following steps:
 - In the **Name** field, enter a name for the certificate.

Create a new certificate

Name [?](#)

Add a description

Public key certificate

[Upload](#)

```
----BEGIN CERTIFICATE----
(paste or upload a public key certificate in .pem format)
----END CERTIFICATE----
```

Certificate chain

[Upload](#)

```
----BEGIN CERTIFICATE----
(paste or upload a certificate chain in .pem format)
----END CERTIFICATE----
```

Private key

[Upload](#)

```
----BEGIN PRIVATE KEY----
(paste or upload a private key in .pem format)
----END PRIVATE KEY----
```

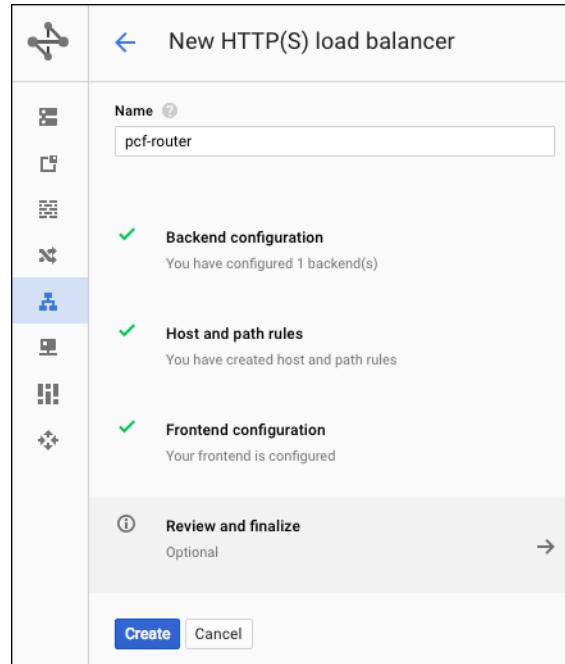
- In the **Public key certificate** field, copy in the contents of your public certificate, or upload your certificate as a .pem file.
- In the **Certificate chain** field, enter or upload your certificate chain in the .pem format. If you are using a self-signed certificate, you do not need to populate this field.
- In the **Private key** field, copy in the contents or upload the .pem file of the private key for the certificate.

16. Review the completed frontend configuration.

Frontend configuration

Specify an IP address, port and protocol. This IP address is the frontend IP for your clients requests. For SSL, a certificate must also be assigned.

Protocol	IP	Port	Certificate
HTTP	192.168.1.10	80	<input type="button" value="X"/>
HTTPS	192.168.1.10	443	<input type="button" value="X"/>
+ Add frontend IP and port			



17. Click **Review and finalize** to verify your configuration.

18. Click **Create**.

Create the 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 **Networking > 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**. Under **Connection termination**, select **No (TCP)**.

Please answer a few questions to help us select the right load balancing type for your application.

Internet facing or internal only

Do you want to load balance traffic from the Internet to your VMs or only between VMs in your network?

- From Internet to my VMs
- Only between my VMs

Connection termination

Do you want to offload SSL processing to the Load Balancer?

- Yes (SSL Proxy)
- No (TCP)

Continue

4. Click **Continue**.

5. In the **New TCP load balancer** window, enter `pcf-websockets` in the **Name** field.

Backend configuration

Name `pcf-websockets`

Region `us-central1`

Backends `Select existing instance groups` `Select existing instances`

Backup pool `(Optional)` `None`

Failover ratio `10` %

Health check `health`
port: 8080, timeout: 5s, check interval: 5s, unhealthy threshold: 2 attempts

Session affinity `None`

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 Subnet](#).
- From the **Health check** drop-down menu, select the [Health check](#) that you created above. 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. Enter a **Name** for the new static IP address and an optional description. For example, `pcf-websockets-ip`.
3. Click **Reserve**.

Frontend configuration

Specify an IP address, port and protocol. This IP address is the frontend IP for your clients requests.

Protocol	IP	Port
TCP	104.131.125.138 (pcf-websoc...)	443
+ Add frontend IP and port		

- Port: 443

8. Click **Review and finalize** to verify your configuration.

New TCP load balancer	Review and finalize				
<p>Name ? pcf-websockets</p> <p>Backend configuration Your backend is configured</p> <p>Frontend configuration Your frontend is configured</p> <p>Review and finalize <small>Optional</small></p> <p>Create Cancel</p>	<p>Backend Name: pcf-websockets Region: us-central1 Session affinity: None Health check: health</p> <p>Frontend</p> <table border="1"> <thead> <tr> <th>Protocol</th> <th>IP:Port</th> </tr> </thead> <tbody> <tr> <td>TCP</td> <td>104.131.125.138:443</td> </tr> </tbody> </table>	Protocol	IP:Port	TCP	104.131.125.138:443
Protocol	IP:Port				
TCP	104.131.125.138:443				

9. Click **Create**.

Create the SSH Proxy Load Balancer

1. From the GCP Console, select **Networking > Load Balancing > Create load balancer**

TCP Load Balancing

Layer 4 load balancing or proxy for applications that rely on TCP/SSL protocol [Learn more](#)

Configure
TCP LB
SSL Proxy

Options
Internet-facing only
Single or multi-region

[Start configuration](#)

2. Under **TCP Load Balancing**, click **Start configuration**.

3. Under **Internet facing or internal only**, select **From Internet to my VMs**.

Please answer a few questions to help us select the right load balancing type for your application.

Internet facing or internal only

Do you want to load balance traffic from the Internet to your VMs or only between VMs in your network?

From Internet to my VMs
 Only between my VMs

Connection termination

Do you want to offload SSL processing to the Load Balancer?

Yes (SSL Proxy)
 No (TCP)

Continue

4. Under **Connection termination**, select **No (TCP)**.

5. Click **Continue**.

New TCP load balancer

Name

Backend configuration
 You have not configured your backend yet

Frontend configuration
 You have not configured your frontend yet

Review and finalize
 Optional

Create **Cancel**

6. In the **New TCP load balancer** window, enter **pcf-ssh** 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**:
- **Failover ratio**:
- **Health check**:

The screenshot shows the configuration steps for a load balancer:

- Backend configuration:** Your backend is configured.
- Frontend configuration:** Your frontend is configured.
- Review and finalize:** Optional.

Configuration details on the right:

- Name:** pcf-ssh
- Region:** us-west1
- Backends:** Select existing instance groups (dropdown: No instance groups in this region)
- Backup pool (Optional):** None
- Failover ratio:** (empty input field)
- Health check:** No health check
- Session affinity:** None

Buttons at the bottom: **Update** and **Cancel**.

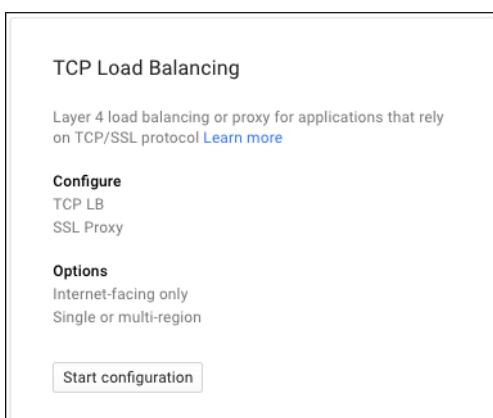
8. Select **Frontend configuration**, and add the following:
 - **Protocol:** `TCP`
 - **IP:** Select the IP address that you created in [Create the TCP WebSockets Load Balancer](#).
 - **Port:** `2222`
9. Optionally, review and finalize your load balancer.
10. Click **Create**.

(Optional) Create the 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 **Networking > Load Balancing > Create load balancer**
2. Under **TCP Load Balancing**, click **Start configuration**.



3. Under **Connection termination**, select **No (TCP)**. Click **Continue**.

Load balancing

Please answer a few questions to help us select the right load balancing type for your application.

Internet facing or internal only

Do you want to load balance traffic from the Internet to your VMs or only between VMs in your network?

- From Internet to my VMs
- Only between my VMs

Connection termination

Do you want to offload SSL processing to the Load Balancer?

- Yes (SSL Proxy)
- No (TCP)

Continue

4. On the **New TCP load balancer** screen, enter a unique name for the load balancer in the **Name** field. For example, `pcf-tcp-router`.

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](#).
- **Health check**: Select the health check you created with the [TCP WebSockets load balancer](#)

- Click **Save and continue**.

New TCP load balancer		Backend configuration
Name ? <input type="text" value="pcf-tcp-router"/>	Name ? <input type="text" value="pcf-tcp-router"/>	
✓ Backend configuration Your backend is configured	Region ? <input type="text" value="us-central1"/>	
✓ Frontend configuration Your frontend is configured	Backends ? <input type="button" value="Select existing instance groups"/> <input type="button" value="Select existing instances"/> No instance groups in this region	
i Review and finalize Optional	Backup pool ? (Optional) <input type="text" value="None"/>	
	Failover ratio ? <input type="text" value="10"/> %	
	Health check ? <input type="text" value="health"/> port: 8080, timeout: 5s, check interval: 5s, unhealthy threshold: 2 attempts	
	Session affinity ? <input type="text" value="None"/>	
<input type="button" value="Create"/> <input type="button" value="Cancel"/>		

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, `pcf-tcp-router-ip`.
3. Click **Reserve**.

- **Port**: `1024-65535`

<p>← New TCP load balancer</p> <p>Name <small>?</small> <input type="text" value="pcf-tcp-router"/></p> <p>✓ Backend configuration Your backend is configured</p> <p>✓ Frontend configuration Your frontend is configured →</p> <p>ⓘ Review and finalize Optional</p> <p>Create Cancel</p>	<p>Frontend configuration</p> <p>Specify an IP address, port and protocol. This IP address is the frontend IP for your clients requests.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%;">Protocol</th> <th style="width: 40%;">IP</th> <th style="width: 40%;">Port</th> </tr> </thead> <tbody> <tr> <td>TCP</td> <td>104.192.170.115 (Jewell's add...)</td> <td>1024-65000</td> </tr> <tr> <td colspan="3" style="text-align: right;">+ Add frontend IP and port</td> </tr> </tbody> </table>	Protocol	IP	Port	TCP	104.192.170.115 (Jewell's add...)	1024-65000	+ Add frontend IP and port		
Protocol	IP	Port								
TCP	104.192.170.115 (Jewell's add...)	1024-65000								
+ Add frontend IP and port										

7. Click **Review and finalize** to verify your configuration.

8. Click **Create**.

What to Do Next

- (Optional) To save time during the final stage of the installation process, you can start downloading the Elastic Runtime tile. See[Step 1: Download the Elastic Runtime Tile](#) of the *Deploying Elastic Runtime on GCP* topic.
- Proceed to the next step in the deployment, [Launching an Ops Manager Director Instance on GCP](#).

Launching an Ops Manager Director Instance on 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 Elastic Runtime 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 **Pivotal Cloud Foundry Ops Manager for GCP**. When you click on the download link, your browser downloads or opens the [OpsManager_version_onGCP.pdf](#) file.

This PDF document provides 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, `om-pcf`.
 - **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](#).

The screenshot shows the 'Create an image' dialog in the GCP Console. On the left is a sidebar with icons for Compute Engine, Storage, and other services. The main area has a title 'Create an image' with a back arrow. It contains the following fields:

- Name:** om-pcf
- Family (Optional):** (empty field)
- Description (Optional):** (empty field)
- Encryption:** Automatic (recommended)
- Source:** Cloud Storage file
- Cloud Storage file:** A dropdown menu with one item selected: 'sr-images/pivotal-ops-manager-20160116191000-541000.tar.gz'. Below this is a 'Browse' button.

At the bottom are 'Create' and 'Cancel' buttons, and a note 'Equivalent REST or command line'.

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.

Name	Size	Created by	Family	Creation time
<input checked="" type="checkbox"/> om-pcf	50 GB	CF-Docs		Nov 22, 2016, 10:07:54 AM

2. Click **Create Instance**.

3. In the **Create an instance form**, complete the following fields:

- **Name:** Enter a name, such as `om-pcf-1a`.
- **Zone:** Choose a zone from the region in which you created your network.
- **Boot disk:** Click **Change**, then perform the following steps:
 - Click **Custom images** if it is not already selected.
 - Select the Ops Manager image you created in the previous step if it is not already selected.

Boot disk				
Select an image or snapshot to create a boot disk; or attach an existing disk.				
OS images	Application images	Custom images	Snapshots	
Existing disks				
<input checked="" type="radio"/> om-pcf Created from CF-Docs on Oct 18, 2016, 11:38:02 AM				

- 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:** Select this checkbox.

Name	om-pcf-1a
Zone	us-central1-b
Machine type	1 vCPU
Boot disk	New 50 GB standard persistent disk Image om-pcf
Identity and API access	Service account: bosh Access scopes: Use IAM roles with service accounts to control VM access Learn more
Firewall	Add tags and firewall rules to allow specific network traffic from the Internet <input checked="" type="checkbox"/> Allow HTTP traffic <input checked="" type="checkbox"/> Allow HTTPS traffic

- **Allow HTTPS traffic:** Select this checkbox.
- **Management:** Click **Management, disk, networking, SSH keys** and select the **Management** tab. In the **Tags** field, enter

[pcf-opsmanager]. This tag applies the firewall rule you created in [Create Firewall Rules for the Network](#) to the Ops Manager VM.

Create an instance

Management Disks Networking SSH Keys

Description (Optional)

Tags (Optional) pcf-opsmanager

Automation

Startup script (Optional)
You can choose to specify a startup script that will run when your instance boots up or restarts. Startup scripts can be used to install software and updates, and to ensure that services are running within the virtual machine. [Learn more](#)

Metadata (Optional)
You can set custom metadata for an instance or project outside of the server-defined metadata. This is useful for passing in arbitrary values to your project or instance that can be queried by your code on the instance. [Learn more](#)

Key Value [Add item](#)

Availability policy

Preemptibility
A preemptible VM costs much less, but lasts only 24 hours. It can be terminated sooner due to system demands. [Learn more](#)

Off (recommended)

Automatic restart
Compute Engine can automatically restart VM instances if they are terminated for non-user-initiated reasons (maintenance event, hardware failure, software failure, etc.)

On (recommended)

On host maintenance
When Compute Engine performs periodic infrastructure maintenance it can migrate your VM instances to other hardware without downtime.

Migrate VM instance (recommended)

[Less](#)

- Networking: Select the **Networking** tab, and perform the following steps:

- For **Network** and **Subnetwork**, select the network and subnetwork 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 **Internal IP**, select **Custom**. Enter **10.0.0.4** as the **Internal IP address**. This internal IP address should be located within the reserved IP range that you will [configure in 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.

Firewall Add tags and firewall rules to allow specific network traffic from the Internet

Allow HTTP traffic
 Allow HTTPS traffic

Management Disks **Networking** SSH Keys

Network opsmgr

Subnetwork central

Internal IP Custom
10.0.0.4

External IP Ephemeral
None
pcf-lb-address (146.148.51.195)
New static IP address...

[Less](#)

- **SSH Keys:** Select the **SSH Keys** tab, and perform the following steps.

- Generate an SSH key for the `ubuntu` user. Creating the `ubuntu` user allows you to directly access the Ops Manager VM using SSH. For example, on your local machine, open a terminal and type:

```
$ ssh-keygen -t rsa -f ubuntu-key -C ubuntu@local
```

Press enter twice to provide no passphrase. This command outputs a private key and a public key, in this example `ubuntu-key.pub`.

- Open the public key file. Then copy and paste the public key data including the username at the end `ubuntu@local`, into the **key data** field. The form then adds an `ubuntu` SSH entry with the username `ubuntu` automatically populated for you.

Management Disks Networking **SSH Keys**

These keys allow access only to this instance, unlike [project-wide SSH keys](#)
[Learn more](#)

Block project-wide SSH keys
When checked, project-wide SSH keys cannot access this instance [Learn more](#)

ubuntu	ubuntu:ssh-rsa AA... BB... CC... DD... EE... FF... GG... HH... II... JJ... KK... LL... MM... NN... OO... PP... QQ... RR... SS... TT... UU... VV... WW... XX... YY... ZZ... PKEY:RSA-4096
--------	--

[+ Add item](#)

- Store the private key in a secure location.

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) to the `om-public-ip` 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.

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.

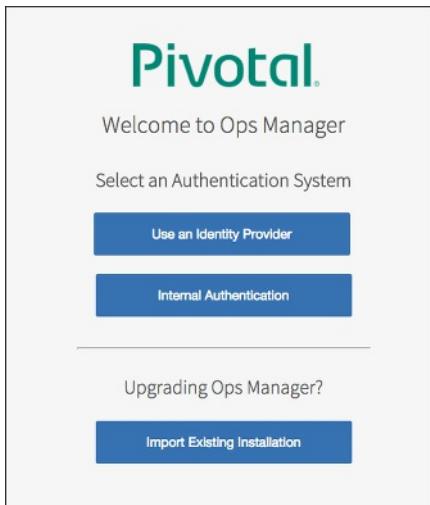
Step 1: Access Ops Manager

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 are using 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](./pcf-director-proxy-settings.html) topic.
 - Read the **End User License Agreement**, and select the checkbox to accept the terms.
 - Click **Setup Authentication**.

Pivotal

Internal Authentication

Username

Password

Password confirmation

Decryption passphrase

Decryption passphrase confirmation

Http proxy

Https proxy

No proxy

I agree to the terms and conditions of the [End User License Agreement](#).

Setup Authentication

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

Pivotal.

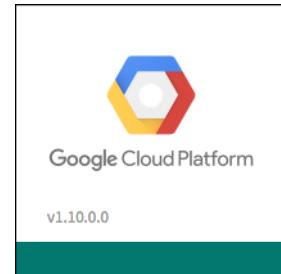
Welcome!

Email

Password

SIGN IN

Step 2: Google Cloud Platform Config



- Click the **Google Cloud Platform** tile within the **Installation Dashboard**.

- 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 `pcf-vms`.
- Select an authentication mechanism for the Ops Manager VM:

- The Ops Manager VM Service Account** Select this 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.

- **AuthJSON:** As an alternative, select this option, 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.

The screenshot shows a configuration dialog titled "Google Cloud Platform Config". It contains fields for "Project ID*" (containing "your-gcp-project-id"), "Default Deployment Tag" (containing "pcf-vms"), and two radio button options: "The Ops Manager VM Service Account" (selected) and "AuthJSON". A large text area for pasting JSON content is empty. At the bottom is a blue "Save" button.

3. Click **Save**.

Step 3: Director Config Page

1. Select **Director Config** to open the **Director Config** page.

Director Config

NTP Servers (comma delimited)*



Metrics IP Address

Enable VM Resurrector Plugin

Enable Post Deploy Scripts

Recreate all VMs

This will force BOSH to recreate all VMs on the next deploy. Persistent disk will be preserved.

Enable bosh deploy retries

This will attempt to re-deploy a failed deployment up to 5 times.

Keep Unreachable Director VMs

2. In the **NTP Servers (comma delimited)** field, enter .
3. (Optional) If you are using [JMX Bridge](#), enter your **Metrics IP Address**.
4. Select the **Enable VM Resurrector Plugin** checkbox to enable the Ops Manager Resurrector functionality and increase Elastic Runtime availability.
5. 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.
6. Select **Recreate all VMs** to force BOSH to recreate all VMs on the next deploy. This process does not destroy any persistent disk data.
7. Select **Enable bosh deploy retries** if you want Ops Manager to retry failed BOSH operations up to five times.
8. Select **Keep Unreachable Director VMs** if you want to preserve Ops Manager Director VMs after a failed deployment for troubleshooting purposes.
9. (Optional) Select **HM Pager Duty Plugin** to enable Health Monitor integration with PagerDuty.

HM Pager Duty Plugin

Service Key*

YOUR-PAGERDUTY-SERVICE-KEY

HTTP Proxy

YOUR-HTTP-PROXY

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

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

HM Email Plugin

Host*

smtp.example.com

Port*

25

Domain*

cloudfoundry.example.com

From*

user2@example.com

Recipients*

user@example.com, user1@example.com

Username

user

Password

.....

Enable TLS

- **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.

11. For **Blobstore Location**, select **Internal**.

12. For **Database Location**, select **Internal**.

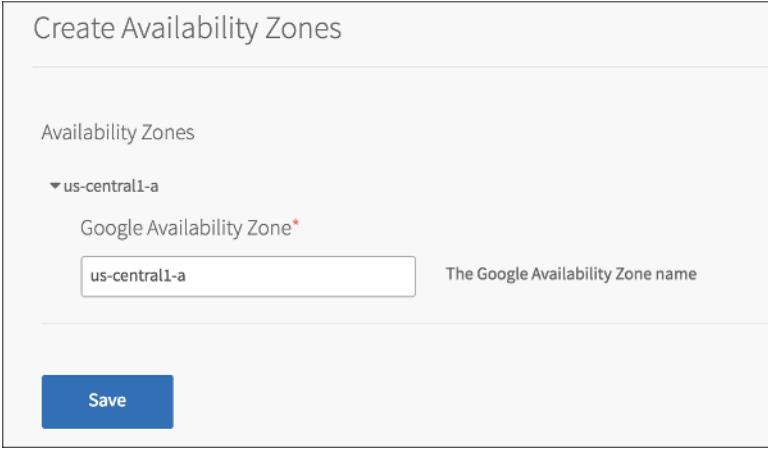
13. **Max Threads** sets the maximum number of threads that the Ops Manager Director can run simultaneously. Pivotal recommends that you leave the field blank to use the default value, unless doing so results in rate limiting or errors on your IaaS.

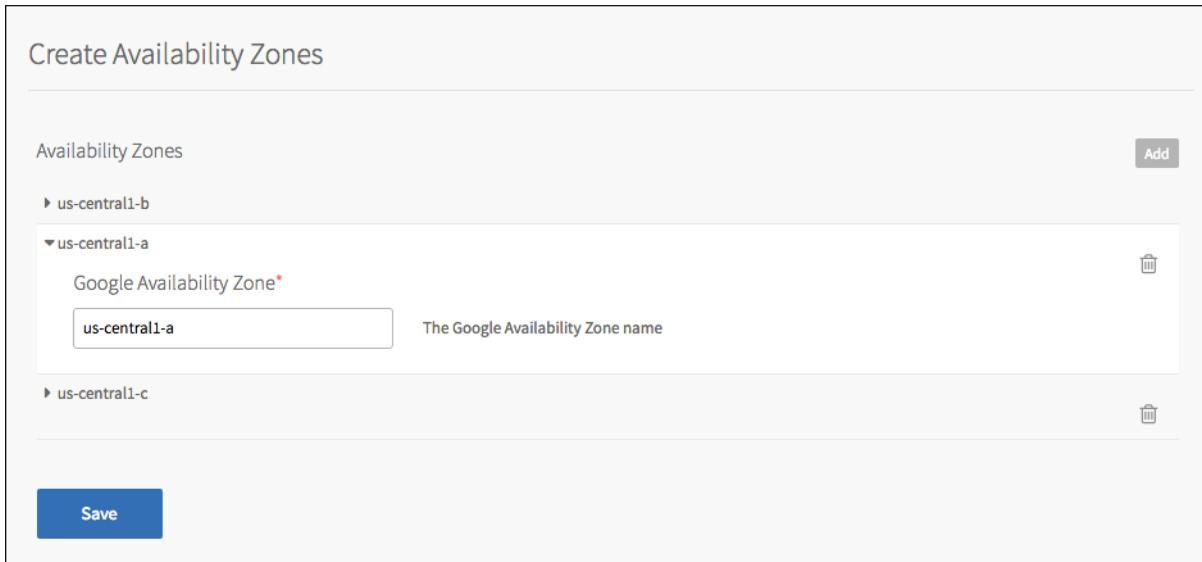
14. (Optional) To add a custom URL for your Ops Manager Director, enter a valid hostname in **Director Hostname**. You can also use this field to configure [a load balancer in front of your Ops Manager Director](#).

15. Click **Save**.

Step 4: Create Availability Zones Page

Note: Pivotal recommends at least three availability zones for a highly available installation of Elastic Runtime. 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 a single availability zone deployment, in **Google Availability Zone**:
 - Enter the zone you associated to the backend service instance group of the HTTP(S) Load Balancer. For example, if you are using the `us-central1` region and selected `us-central1-a` as the zone for your HTTP(S) Load Balancer instance group, enter `us-central1-a`.
- 
- The screenshot shows a 'Create Availability Zones' form. It has a header 'Create Availability Zones'. Below it is a section titled 'Availability Zones' with a dropdown menu showing 'us-central1-a'. Underneath is a field labeled 'Google Availability Zone*' containing 'us-central1-a'. To the right of this field is the placeholder text 'The Google Availability Zone name'. At the bottom is a blue 'Save' button.
- Click **Save**.
 4. For a multiple availability zone deployment, in **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](#).



The screenshot shows a 'Create Availability Zones' form with multiple availability zones listed. At the top right is an 'Add' button. Below it is a table-like structure with columns for 'Availability Zones' and 'Delete'. The first row shows 'us-central1-b'. The second row shows 'us-central1-a' with a 'Delete' icon to its right. The third row shows 'us-central1-c'. At the bottom is a blue 'Save' button.

- Click **Save**.
5. 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 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.

 **Note:** Do not select the **Services Networks** checkbox when configuring Ops Manager for the first time. If selected, the network may not appear as a drop-down option in the [Assign AZ and Networks Page](#)

- Under **Subnets**, complete the following fields:
 - **Google Network Name:** Enter the network, subnet and region names of the Google Network you created in [Preparing to Deploy PCF on GCP](#). The format is `NETWORK-NAME/SUBNET-NAME/REGION-NAME`. For example, `opsmgr/central/us-central1`.
 - **CIDR:** Enter `10.0.0.0/20`. Ops Manager deploys VMs to this CIDR block.
 - **Reserved IP Ranges:** Enter `10.0.0.1-10.0.0.9`. Ops Manager avoids deploying VMs to any IP address in this range.
 - **DNS:** Enter `169.254.169.254, 8.8.8.8`.

 **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.

- **Gateway:** Enter `10.0.0.1`.
 - **Availability Zones:** Select the availability zone for the subnet. If you created multiple availability zones, select all listed availability zones.
- If you want to add more subnets, click **Add Subnet**.

Create Networks

Warning: Pivotal recommends keeping the IP settings throughout the life of your installation. Ops Manager may prevent you from changing them in the future. Contact Pivotal support for help completing such a change.

Verification Settings

Enable ICMP checks

Networks

[Add Network](#)

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

[opsmgr](#)



Name*

opsmgr



Service Network

Subnets

[Add Subnet](#)

Google Network Name*

opsmgr/opsmgr-subnet/us-central1

CIDR*

10.0.0.0/20

Reserved IP Ranges

10.0.0.1-10.0.0.9

DNS*

169.254.169.254, 8.8.8.8

Gateway*

10.0.0.1

Availability Zones*

us-central1-b

us-central1-a

us-central1-c

[Save](#)

4. Click **Save**.

Step 6: Assign AZs and Networks Page

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

Assign AZs and Networks

The Ops Manager Director is a single instance.

Choose the availability zone in which to place that instance. It is highly recommended that you backup this VM on a regular basis to preserve settings.

Singleton Availability Zone

us-central1-a

Network

opsmgr

Save

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

Security

Trusted Certificates
 -----BEGIN CERTIFICATE-----
 TH...
 -----END CERTIFICATE-----

These certificates enable BOSH-deployed components to trust a custom root certificate.

Generate VM passwords or use single password for all VMs

Generate passwords
 Use default BOSH password

Save

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 Trusted Registries for running app instances in Docker containers, enter the certificate for your private Docker Trusted Registry in this field. See the [Using Docker Trusted 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: Resource Config Page

1. Select **Resource Config**.

2. Ensure that the **Internet Connected** checkboxes are selected for all jobs. This gives all VMs a public IP address that enables outbound Internet access.

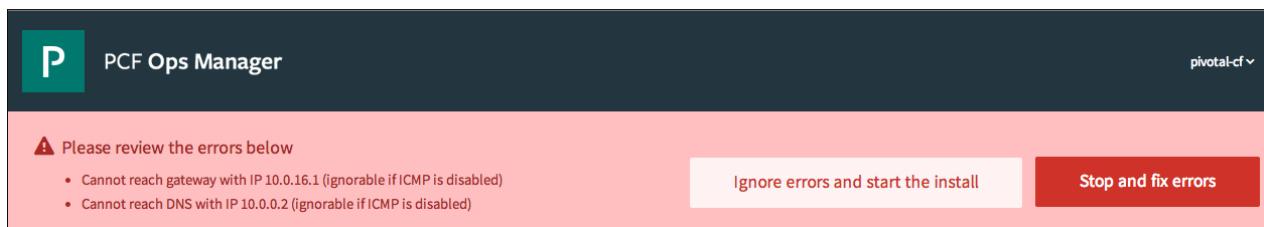
Note: If you want to provision a Network Address Translation (NAT) box to provide Internet connectivity to your VMs instead of providing them with public IP addresses, deselect the **Internet Connected** checkboxes. For more information about using NAT in GCP, see the [GCP documentation](#).

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.

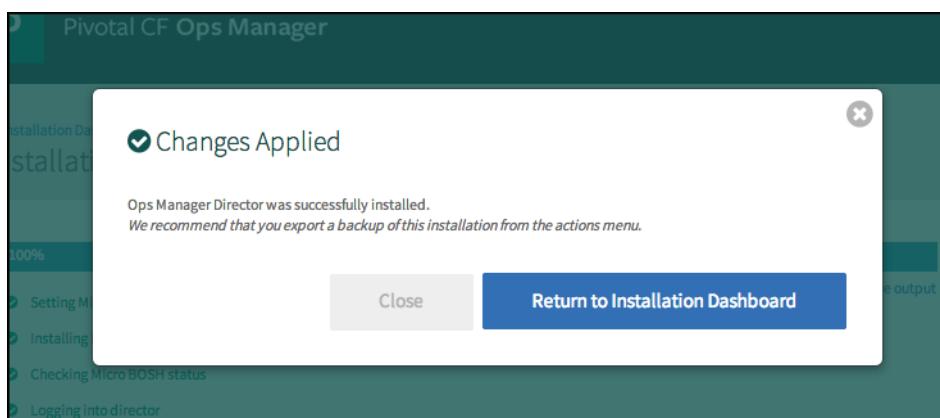
Note: If you set a field to **Automatic** and the recommended resource allocation changes in a future version, Ops Manager automatically uses the updated recommended allocation.

Step 9: 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 Elastic Runtime on GCP](#) topic.

Deploying Elastic Runtime on GCP

Page last updated:

This topic describes how to install and configure Elastic Runtime for Pivotal Cloud Foundry (PCF) on Google Cloud Platform (GCP).

Before beginning this procedure, ensure that you have successfully completed the [Configuring Ops Manager Director on GCP](#) topic.

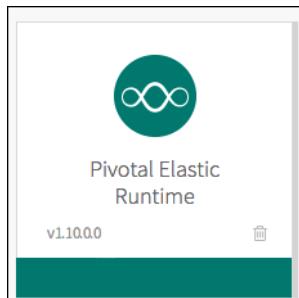
Note: If you plan to [install the PCF IPsec add-on](#), you must do so before installing any other tiles. Pivotal recommends installing IPsec immediately after Ops Manager, and before installing the Elastic Runtime tile.

Step 1: Download the Elastic Runtime Tile

1. If you have not already downloaded Elastic Runtime, log in to [Pivotal Network](#), and click on **Pivotal Cloud Foundry Elastic Runtime**.
2. From the **Releases** drop-down, select the release to install.
3. Click on **PCF Elastic Runtime** to download the Elastic Runtime .pivotal file

Step 2: Add Elastic Runtime to Ops Manager

1. Navigate to the Pivotal Cloud Foundry Operations Manager Installation Dashboard.
2. Click **Import a Product** to add the Elastic Runtime tile to Ops Manager. This may take a while depending on your connection speed.
3. On the left, click the plus icon next to the imported Elastic Runtime product to add it to the Installation Dashboard.
4. Click the newly added **Elastic Runtime** 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 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 Elastic Runtime.

Note: For production deployments, Pivotal recommends at least three Availability Zones for a highly available installation of Elastic Runtime.

AZ and Network Assignments

- Assign AZs and Networks
- Domains
- Networking
- Application Containers
- Application Developer Controls
- Application Security Groups
- Authentication and Enterprise SSO
- Databases

Place singleton jobs in
 first-az

Balance other jobs in
 first-az

Network

Save

- 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.

- Locate the static IP addresses of the load balancers you created in [Preparing to Deploy PCF on GCP](#):
 - An HTTP(S) load balancer named `pcf-router`
 - A TCP load balancer for WebSockets named `pcf-websockets`
 - A TCP load balancer named `pcf-ssh`
 - A TCP load balancer for the TCP router if you plan on enabling the TCP routing feature



You can locate the static IP address of each load balancer by clicking its name under **Networks > Load balancing** in the GCP Console.

- Log in to the DNS registrar that hosts your domain. Examples of DNS registrars include Network Solutions, GoDaddy, and Register.com.
- 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	<code>pcf-router</code>	Yes	*.system.example.com
*.YOURAPPSDOMAIN	<code>pcf-router</code>	Yes	*.apps.example.com
doppler.YOURSYSTEMDOMAIN	<code>pcf-websockets</code>	Yes	doppler.system.example.com
loggregator.YOURSYSTEMDOMAIN	<code>pcf-websockets</code>	Yes	loggregator.system.example.com
ssh.YOURSYSTEMDOMAIN	<code>pcf-ssh</code>	Yes, to allow SSH access to apps	ssh.system.example.com
tcp.YOURDOMAIN	IP address of the TCP load balancer for TCP routing	No, only set up if you have enabled the TCP routing feature	tcp.example.com

- 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**.

Elastic Runtime hosts applications at subdomains under its apps domain and assigns system components to subdomains under its system domain. You need to configure a wildcard DNS for both the apps domain and system domain. The two domains can be the same, although this is not recommended.

System Domain *

This domain is for system-level PCF components, such as Apps Manager, service brokers, etc. You must set up a wildcard DNS record for this domain that points to your entry point load balancer or HAProxy.

Apps Domain *

Save

2. Enter the system and application domains.

- The **System Domain** defines your target when you push apps to Elastic Runtime.
- The **Apps Domain** defines where Elastic Runtime 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 Elastic Runtime later on in the installation process. See the [Configure Load Balancers](#) section of this topic for more information.

3. Under **Select one of the following point-of-entry options** choose the **Forward SSL to Elastic Runtime Router** option.

 **Note:** As a clarification, GCP load balancers actually forward both encrypted (WebSockets) and unencrypted (HTTP and TLS-terminated HTTPS) traffic to the Elastic Runtime Router (Gorouter). This point-of-entry selection accommodates this specific characteristic of GCP deployments. For more details on network topology in GCP, see [Reference Architecture for Pivotal Cloud Foundry on GCP](#).

4. Complete the fields for the **Router SSL Termination Certificate** and **Private Key** and **Router SSL Ciphers**. For details about providing SSL termination certificates and keys, see the [Providing a Certificate for your SSL Termination Point](#) topic.

Select one of the following point-of-entry options:^{*}

Forward SSL to Elastic Runtime Router. Assumes an external load balancer is configured to forward encrypted traffic.

Router SSL Termination Certificate and Private Key *



Router SSL Ciphers

Change
Required for SSL encryption between the load balancer and router(s).

Forward unencrypted traffic to Elastic Runtime Router. Assumes an external load balancer is configured to forward unencrypted traffic.
 Forward SSL to HA Proxy. Like first option - Assumes an external load balancer is configured to forward encrypted traffic.

5. 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.

6. Select the **Disable insecure cookies on the Router** checkbox to set the secure flag for cookies generated by the router.
7. 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](#).

Disable SSL certificate verification for this environment

Disable insecure cookies on the Router

Enable Zipkin tracing headers on the router

8. 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.
9. The **Loggregator Port** defaults to [443](#) if left blank. Enter a new value to override the default.
10. (Optional) Use the **Applications Subnet** field if you need to avoid address collision with a third-party service on the same subnet as your apps. Enter a CIDR subnet mask specifying the range of available IP addresses assigned to your app containers. The IP range must be different from the network used by the system VMs.
11. (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.

Choose whether to enable route services. Route services enable you to proxy requests to your app over TLS to arbitrary URLs before hitting your app.*

- Enable route services
- Disable route services

Loggregator Port

Applications Subnet (Only change this if you need to avoid address collision with a third-party service on the same subnet.) *

10.254.0.0/22

Applications Network Maximum Transmission Unit (MTU) (in bytes) *

1454

12. (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.

13. (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.

Router Timeout to Backends (in seconds) (min: 1) *

900

Load Balancer Unhealthy Threshold *

0

Load Balancer Healthy Threshold *

20

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

Router Max Idle Keepalive Connections (min: 0, max: 50000) *

0

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

- a. Select the **Enable TCP Routing** radio button.
- 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.

Enable TCP requests to your apps via specific ports on the TCP router. You will want to configure a load balancer to forward these TCP requests to the TCP routers. If you do not have a load balancer, then you can also send traffic directly to the TCP router.*

- Select this option if you prefer to enable TCP Routing at a later time
- Enable TCP Routing

TCP Routing Ports (one-time configuration, if you want to update this value you can via the CF CLI) *

1024-1123

- 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 Elastic Runtime. See [Configure Load Balancers](#) section of this topic.
- 16. To disable TCP routing, select the **Select this option if you prefer to enable TCP Routing at a later time** radio button. For more information, see the [Configuring TCP Routing in Elastic Runtime](#) topic.
- 17. Click **Save**.

Step 7: Configure Application Containers

1. Select **Application Containers**.

Enable microservice frameworks, private Docker registries, and other services that support your applications at a container level.

Enable Custom Buildpacks

Allow SSH access to app containers

Enable SSH when an app is created

Private Docker Insecure Registry Whitelist

Docker Images Disk-Cleanup Scheduling on Cell VMs*

Never clean up Cell disk-space

Routinely clean up Cell disk-space

Clean up disk-space once threshold is reached

Threshold of Disk-Used (MB) (min: 1) *

Max Inflight Container Starts *

Save

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 deselecting 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 Elastic Runtime 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 Trusted Registries](#) topic for more information.
6. Select your preference for **Docker Images Disk-Cleanup on Cell VMs**. If you choose **Clean up disk-space once threshold is reached** enter a **Threshold of Disk-Used** in megabytes.
7. Optionally, 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. Click **Save**.

Step 8: Configure Application Developer Controls

1. Select **Application Developer Controls**.

Configure restrictions and default settings for applications pushed to Elastic Runtime.

Maximum File Upload Size (MB) (min: 1024, max: 2048) *

Default App Memory (MB) (min: 64, max: 2048) *

Default App Memory Quota per Org (min: 10240, max: 102400) *

Maximum Disk Quota per App (MB) (min: 512, max: 10240) *

Default Disk Quota per App (MB) (min: 512, max: 10240) *

Default Service Instances Quota per Org (min: 0, max: 1000) *

2. Enter your intended maximum file upload size.
3. Enter your default RAM memory allocation per app.
4. Enter your default total RAM memory (RAM) quota per Org. You can change this in the CLI.
5. Enter your maximum and default disk quotas per app.
6. Enter your default service instances quota per Org. You can change this in the CLI.
7. Click **Save**.

Step 9: Review Application Security Groups

Setting appropriate [Application Security Groups](#) is critical for a secure deployment. Type in the box to acknowledge that once the Elastic Runtime deployment completes, you will review and set the appropriate application security groups.

Setting appropriate Application Security Groups that control application network policy is the responsibility of the Elastic Runtime administration team. Please refer to the Application Security Groups topic in the Pivotal Cloud Foundry documentation for more detail on completing this activity after the Elastic Runtime deployment completes.

Type X to acknowledge that you understand this message *

Save

Step 10: Configure Authentication and Enterprise SSO

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

Configure your user store access, which can be an internal user store (managed by Cloud Foundry's UAA) or an external user store (LDAP or SAML). You can also adjust the lifetimes of authentication tokens.

Configure your UAA user account store with either internal or external authentication mechanisms*

- Internal UAA (provided by Elastic Runtime; configure your password policy below)

Minimum Password Length *

Minimum Uppercase Characters Required for Password *

Minimum Lowercase Characters Required for Password *

Minimum Numerical Digits Required for Password *

Minimum Special Characters Required for Password *

Number of Months Before Password Expires *

Maximum Password Entry Attempts Allowed *

- SAML Identity Provider

- LDAP Server

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 Elastic Runtime](#) 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 Elastic Runtime](#) topic.

3. (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.
4. (Optional) Customize the text prompts used for username and password from the cf CLI and Apps Manager login popup.
5. (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 public IP address(es), append a regular expression or regular expressions to match the public IP address(es).

The screenshot shows a configuration form with the following fields:

- Apps Manager Access Token Lifetime (in seconds)**: Value 1209600
- Apps Manager Refresh Token Lifetime (in seconds)**: Value 1209600
- Cloud Foundry CLI Access Token Lifetime (in seconds)**: Value 7200
- Cloud Foundry CLI Refresh Token Lifetime (in seconds)**: Value 1209600. A tooltip indicates: "Set the lifetime of the refresh token for the Cloud Foundry CLI."
- Customize Username Label (on login page)**: Value Email
- Customize Password Label (on login page)**: Value Password
- Proxy IPs Regular Expression**: Value `10\.\d{1,3}\.\d{1,3}\.\d{1,3}|192\.\d{1,3}\.\d{1,3}`

A blue **Save** button is at the bottom.

6. Click **Save**.

Step 11: Configure System Databases

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

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**.

Place the databases used by Elastic Runtime components like Cloud Controller and UAA.

Choose the location of your system databases*

- Internal Databases - MySQL and Postgres (the Postgres DBs are not highly-available, but this selection is required if you want to keep your system data from a pre-1.6.0 Elastic Runtime that you upgraded)
- Internal Databases - MySQL (preferred for complete high-availability)
- External Databases (preferred if, for example, you use AWS RDS)

Save

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: 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.

To create your Elastic Runtime databases, perform the following steps:

- 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
```

- SSH in to your Ops Manager using the [Ops Manager FQDN](#) and the username `ubuntu`:

```
ssh ubuntu@OPS_MANAGER_FQDN
```

- 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=RDHOSTNAME --user=RDSUSERNAME --password=RDSPASSWORD
```

- Run the following MySQL commands to create databases for the seven Elastic Runtime components that require a relational database:

```
CREATE database uaa;
CREATE database cedb;
CREATE database notifications;
CREATE database autoscale;
CREATE database app_usage_service;
CREATE database routing;
CREATE database diego;
CREATE database account;
CREATE database nfsvolume;
```

- Type `exit` to quit the MySQL client, and `exit` again to close your connection to the Ops Manager VM.
- In Elastic Runtime, select **Databases**.
- Select the **External Databases** option.

Place the databases used by Elastic Runtime components like Cloud Controller and UAA.

Choose the location of your system databases*

- Internal Databases - MySQL and Postgres (the Postgres DBs are not highly-available, but this selection is required if you want to keep your system data from a pre-1.6.0 Elastic Runtime that you upgraded)
- Internal Databases - MySQL (preferred for complete high-availability)
- External Databases (preferred if, for example, you use AWS RDS)

Hostname *

TCP Port *

App Usage Service database username *

App Usage Service database password *

8. Complete the following fields in Elastic Runtime:

Elastic Runtime Field	Notes
Hostname	Specify the hostname of the database server.
TCP Port	Specify the port of the database server.
DATABASE-NAME database username	Specify a unique username that can access this specific database on the database server.
DATABASE-NAME database password	Specify a password for the provided username.

9. 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.

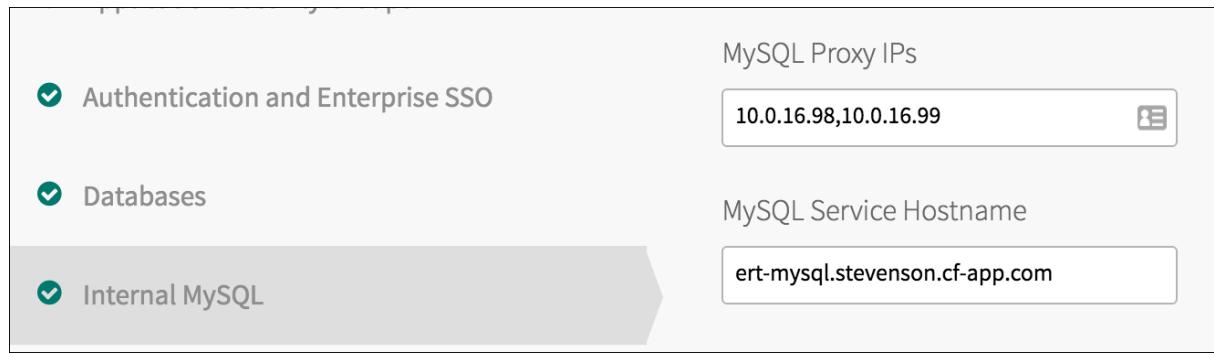
1. Select **Internal MySQL**.
2. In the **MySQL Proxy IPs** field, enter one or more comma-delimited IP addresses that are not in the reserved CIDR range of your network. If a MySQL node fails, these proxies re-route connections to a healthy node. See the [Proxy](#) section of the *MySQL for PCF* topic for more information.
3. (Optional) Configure round-robin DNS to spread requests across your MySQL proxies. Only perform this step if you want to approximate load balancing on your internal MySQL proxies.

- a. Create a DNS [A Record](#) to round robin against your IP addresses. For example:

ert-mysql.stevenson.cf-app.com.	A	10.0.16.98
		10.0.16.99

- b. In the **MySQL Service Hostname** field, enter the hostname you created for round-robin DNS. If you leave this field blank, components are configured with the IP address of the first proxy instance entered in the **MySQL Proxy IPs** field.

 **Caution:** Round-robin DNS does not handle component availability as well as a load balancer. If one or more of the database proxies fail, components that rely on the MySQL database can become unavailable. At time of publication, GCP load balancers only support access to public IP addresses.



4. In the **Replication canary time period** field, leave the default of 30 seconds or modify the value based on your deployment requirements. Lower numbers cause the canary to run more frequently, which adds load to the database.
5. In the **Replication canary read delay** field, leave the default of 20 seconds or increase the value. 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 should send alerts when the cluster experiences a replication issue or when a node is not allowed to auto-rejoin the cluster.

This screenshot shows a configuration panel with three fields:

- Replication canary time period ***: A text input field containing the value '30'.
- Replication canary read delay ***: A text input field containing the value '20'.
- E-mail address (required) ***: A text input field containing the value 'email@example.com'.

7. Under **Automated Backups Configuration**, choose one of three options for MySQL backups:
 - **Disable automatic backups of MySQL**
 - **Enable automated backups from MySQL to an S3 bucket or other S3-compatible file stores** saves your backups to an existing Amazon Web Services (AWS) or [Ceph](#) S3-compatible blobstore.

Automated Backups Configuration*

Disable automated backups of MySQL

Enable automated backups from MySQL to an S3 bucket or other S3-compatible file store

S3 Endpoint URL *

S3 Bucket Name *

Bucket Path *

AWS Access Key ID *

AWS Secret Access Key *

Cron Schedule *

Backup All Nodes

Enable automated backups from MySQL to a remote host via SCP

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 **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 automated backups from MySQL to a remote host via SCP** saves your backups to a remote host using secure copy

Automated Backups Configuration*

Disable automated backups of MySQL

Enable automated backups from MySQL to an S3 bucket or other S3-compatible file store

Enable automated backups from MySQL to a remote host via SCP

Hostname *

Port *

Username *

Private key *

Destination directory *

Cron Schedule *

Backup All Nodes

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 Elastic Runtime tile to 0.

8. 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.

Server Activity Logging*

Disable server activity logging
 Enable server activity logging

Event types *

connect,query

Load Balancer Healthy Threshold *

0

Load Balancer Unhealthy Threshold *

0

Save

9. 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.

10. Click **Save**.

Step 13: Configure File Storage

To minimize system downtime, Pivotal recommends using highly resilient and redundant *external* filestores for your Elastic Runtime file storage.

When configuring file storage for the Cloud Controller in Elastic Runtime, 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 Elastic Runtime tile, select **File Storage**.
2. Select **Internal WebDAV**, and click **Save**.

External Google Cloud Storage

To use external Google file storage for your Elastic Runtime filestore, perform the following steps:

1. Select the **External Google Cloud Storage** option.

This section determines where you would like to place your Elastic Runtime Cloud Controller's file storage.

Configure your Cloud Controller's filesystem*

Internal WebDAV (provided by Elastic Runtime)
 External Google Cloud Storage

Access Key *

[\[copy\]](#)

Secret Key *

[Change](#)

Buildpacks Bucket Name *

Bucket for storing app buildpacks.

Droplets Bucket Name *

Packages Bucket Name *

Resources Bucket Name *

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.

The screenshot shows the Google Cloud Platform Settings page with the 'Interoperability' tab selected. A success message states: 'cf-docs is your default project for interoperable access'. Below this, there is a section for 'Interoperable storage access keys' with an 'Access Key' and 'Secret' field, both of which are blurred. A 'Create a new key' button is visible at the bottom.

- Click **Create a new key**.
 - Copy and paste the generated values into the corresponding Elastic Runtime fields. PCF uses these values for authentication when connecting to Google Cloud Storage.
3. To create buckets in GCP, perform the following steps:
- In the GCP Console, navigate to the **Storage** tab, then click **Create Bucket**.
 - Enter a unique bucket name.
 - For the **Default storage class**, select **Regional**.
 - From the **Regional location** dropdown, select the region associated with your PCF deployment.
 - Click **Create**. When the bucket is created, return to Elastic Runtime to configure the bucket names.
4. For the **Buildpacks Bucket Name**, enter the name of the bucket for storing your app buildpacks.
5. For **Droplets Bucket Name**, enter the name of the bucket for your app droplet storage. Pivotal recommends that you use a unique bucket, but you can use the same bucket as the previous step.
6. For **Resources Bucket Name**, enter the name of the bucket for resources. Pivotal recommends that you use a unique bucket, but you can use the same bucket as the previous step.
7. For **Packages Bucket Name**, enter the name of the bucket for packages. Pivotal recommends that you use a unique bucket, but you can use the same bucket as the previous step.
8. Click **Save**.

Other IaaS Storage Options

[Azure Storage](#) and [External S3-Compatible File Storage](#) are also available as file storage options, but are not recommended for a typical PCF on GCP installation.

Step 14: (Optional) Configure System Logging

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

1. Select **System Logging**.

Configure system logging. Complete the External Syslog fields only if using an external syslogd server.

Enable Cloud Controller security event logging

External Syslog Aggregator Hostname

External Syslog Aggregator Port

The typical syslog port is 514. Ensure syslogd is listening on external interfaces.

External Syslog Network Protocol

Syslog Drain Buffer Size (# of messages) *

Save

2. 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).
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 Elastic Runtime 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**.

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](#) and [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.

Customize colors, images, and text for Apps Manager and the Cloud Foundry login portal.

Company Name



Accent Color

Main Logo (PNGs only)

Square Logo/Favicon (PNGs only)

Footer Text

Defaults to 'Pivotal Software Inc. All rights reserved.'

Add

Footer Links

You may configure up to three links in the Apps Manager footer

Classification Header/Footer Background Color

Classification Header/Footer Text Color

Classification Header Content

Classification Footer Content

Save

2. Click **Save** to save your settings in this section.
3. Select **Apps Manager**. Use this section to configure page names and sidebar links in the **Apps Manager** and **Marketplace** pages.

Configure Apps Manager

Display Marketplace Service Plan Prices

Supported currencies as json *



Define the currency codes and associated symbols (defaults to "[\"usd\": \"\$\", \"eur\": \"€\"]" if blank)

Product Name

Marketplace Name

Customize Sidebar Links
You may configure up to 10 links in the Apps Manager sidebar 

Link	Remove
▶ Marketplace	
▶ Docs	
▶ Tools	

Save

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

Step 16: (Optional) Configure Email Notifications

Elastic Runtime 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**.

Configure Simple Mail Transfer Protocol for the Notifications application to send email notifications about your deployment. This application is deployed as an errand in Elastic Runtime. If you do not need this service, leave this section blank and disable the Notifications and Notifications UI errands.

From Email

Address of SMTP Server

Port of SMTP Server

SMTP Server Credentials

Re

[Change](#)

SMTP Enable Automatic STARTTLS

SMTP Authentication Mechanism*

Plain

SMTP CRAMMD5 secret

[Save](#)

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: (Optional) Add CCDB Restore Key

Perform this step if all of the following are true:

- You deployed Elastic Runtime previously.
- You then stopped Elastic Runtime or it crashed.
- You are re-deploying Elastic Runtime with a backup of your Cloud Controller database.

1. Click **Restore CCDB Encryption Key**.
2. Enter your **Cloud Controller DB Encryption Key**.

You must enter your original database encryption key here once you have deployed Elastic Runtime for the first time. Anytime thereafter your encryption key will remain the same, if you specify the same key here. This is important for backup and restore of data as well as upgrades.

Cloud Controller DB Encryption Key

...

Leave blank unless restoring a Cloud Controller Database exported from a previous installation.

Save

See [Backing Up Pivotal Cloud Foundry](#) for more information.

Step 18: Configure Smoke Tests

The Smoke Tests errand runs basic functionality tests against your Elastic Runtime 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 org and space**, which creates an ad-hoc org and space for running smoke tests and deletes them afterwards. Otherwise, select **Specified org and space** and complete the fields to specify where you want to run smoke tests.

Specify a Cloud Foundry organization and space where smoke tests can run if in the future you delete your Elastic Runtime deployment domains.

Choose whether to create a new org and space on demand for smoke tests*

- Temporary org and space (This org and space are deleted after smoke tests finish.)
 Specified org and space (The org and space must have a domain available for routing.)

Organization *

Space *

Domain *

Save

3. Click **Save**.

Step 19: (Optional) Enable Advanced Features

The Advanced Features section of Elastic Runtime 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

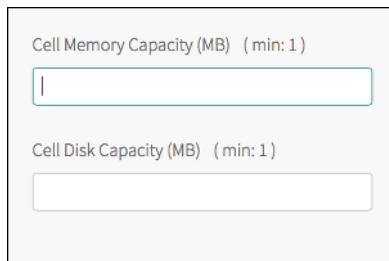
If your apps do not use the full allocation of disk space and memory set in the **Resource Config** tab, you might want use this feature. These fields control the amount to overcommit disk and memory resources to each Diego Cell VM.

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**.



The screenshot shows a form with two input fields. The first field is labeled "Cell Memory Capacity (MB) (min: 1)" with a text input box containing a single vertical bar character. The second field is labeled "Cell Disk Capacity (MB) (min: 1)" with an empty text input box.

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

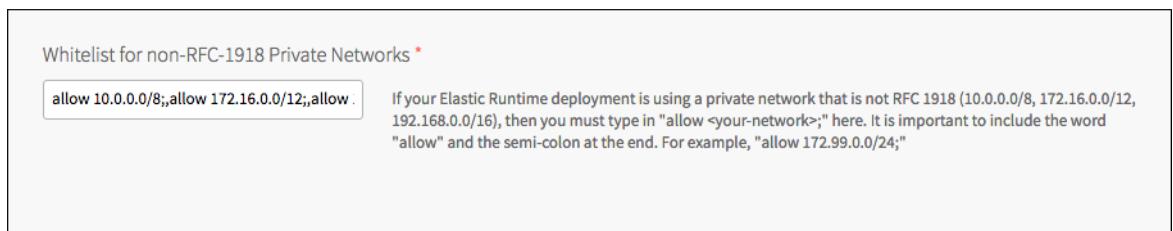
Some private networks require extra configuration so that internal file storage (WebDAV) can communicate with other PCF processes.

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.



The screenshot shows a text input field for the whitelist. The placeholder text is "Whitelist for non-RFC-1918 Private Networks *". Below the input field is a note: "allow 10.0.0.0/8;allow 172.16.0.0/12;allow . If your Elastic Runtime deployment is using a private network that is not RFC 1918 (10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16), then you must type in "allow <your-network>;" here. It is important to include the word "allow" and the semi-colon at the end. For example, "allow 172.99.0.0/24;"

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 Elastic Runtime 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 Elastic Runtime.

To modify the value of the **CF CLI Connection Timeout**, perform the following steps:

1. Select **Advanced Features**.



A screenshot of a web interface showing a single input field with a light gray border. The field contains the number "15". Above the field, the text "CF CLI Connection Timeout" is displayed in a small, dark font.

2. Add a value, in seconds, to the **CF CLI Connection Timeout** field.
3. Click **Save**.

Container-to-Container Networking

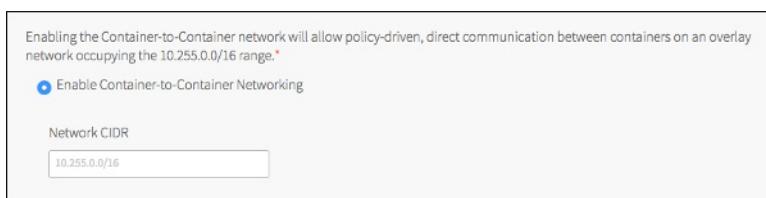
Enabling Container-to-Container Networking allows direct, policy-driven communication between containers on an overlay network occupying a specific range. For more information about Container-to-Container Networking, see the [Administering Cloud Foundry Networking](#) topic.

By default, PCF sets the overlay range to `10.255.0.0/16`, but you can modify this range when you enable Container to Container Networking.

Enable Container-to-Container Networking

To enable Container-to-Container Networking, perform the following steps:

1. Select **Advanced Features**.
2. Select **Enable Container-to-Container Networking**. Optionally, add a range of IP addresses in CIDR format to use instead of the default



A screenshot of a configuration dialog box. At the top, there is a note: "Enabling the Container-to-Container network will allow policy-driven, direct communication between containers on an overlay network occupying the 10.255.0.0/16 range." Below this, there is a radio button labeled "Enable Container-to-Container Networking" which is selected. Underneath, there is a section titled "Network CIDR" with an input field containing the value "10.255.0.0/16".

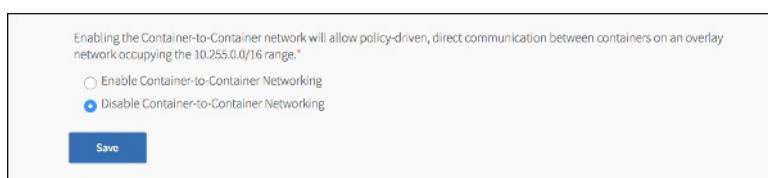
overlay network.

3. Click **Save**.

Disable Container-to-Container Networking

To disable Container-to-Container Networking, perform the following steps:

1. Select **Advanced Features**.
2. Select **Disable Container-to-Container Networking**.



A screenshot of a configuration dialog box. At the top, there is a note: "Enabling the Container-to-Container network will allow policy-driven, direct communication between containers on an overlay network occupying the 10.255.0.0/16 range." Below this, there are two radio buttons: "Enable Container-to-Container Networking" (unchecked) and "Disable Container-to-Container Networking" (checked). At the bottom of the dialog is a blue "Save" button.

3. Click **Save**.

CF API Rate Limiting

Enabling 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.

Enable CF API Rate Limiting

To enable CF API Rate Limiting, perform the following steps:

1. Select **Advanced Features**.
2. Under **Enable CF API Rate Limiting**, select **Enable**.

Enabling CF API Rate Limiting will prevent 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.*

Enable

General Limit *

2000

Unauthenticated Limit *

100

3. 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**.
4. For **Unauthenticated Limit**, enter the number of requests an unauthenticated client is allowed to make over an hour interval. The default value is **100**.
5. Click **Save**.

Disable CF API Rate Limiting

To disable CF API Rate Limiting, perform the following steps:

1. Select **Advanced Features**.
2. Under **Enable CF API Rate Limiting**, select **Disable**.

Enabling CF API Rate Limiting will prevent 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.*

Enable

Disable

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 Elastic Runtime. There are two types of errands: *post-deploy errands* run after the product is installed, and *pre-delete errands* run before the product is 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. The Elastic Runtime 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.

Errands

Errands are scripts that run at designated points during an installation.

Post-Deploy Errands

Smoke Test Errand

Runs Smoke Tests against your Elastic Runtime installation

On

Apps Manager Errand

Pushes the Pivotal Apps Manager application to your Elastic Runtime installation

On

Notifications Errand

Pushes the Pivotal Notifications application to your Elastic Runtime installation

On

Notifications UI Errand

Pushes the Notifications UI component to your Elastic Runtime installation

On

Pivotal Account Errand

Pushes the Pivotal Account application to your Elastic Runtime installation

On

Autoscaling Errand

Pushes the Pivotal App Autoscaling application to your Elastic Runtime installation

On

Autoscaling Registration Errand

Registers the Autoscaling Service Broker

On

NFS Broker Errand

Pushes the NFS Broker application to your Elastic Runtime installation

On

There are no pre-delete errands for this product.

Save

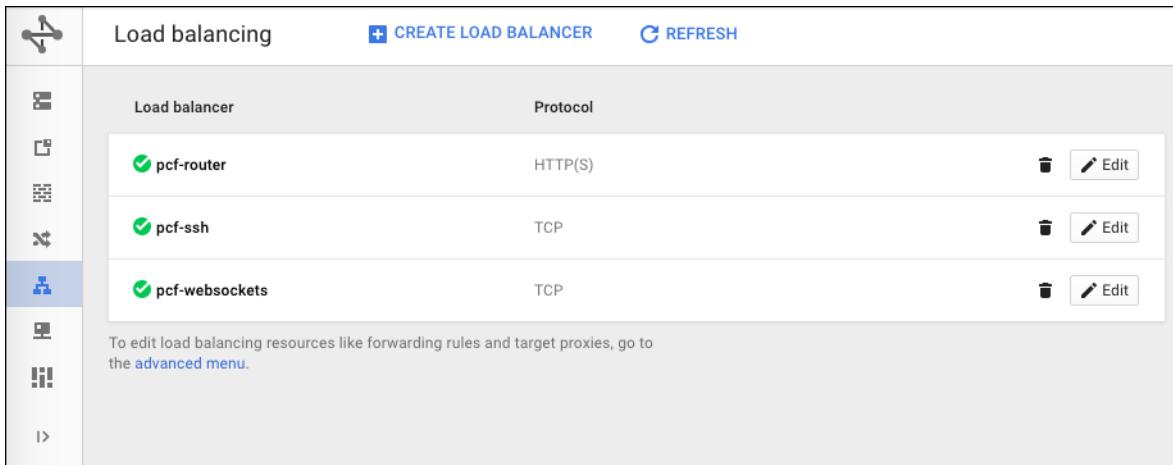
- **Smoke Test Errand** verifies that your deployment can do the following:
 - Push, scale, and delete apps
 - Create and delete orgs and spaces
- **Apps Manager Errand** deploys the Apps Manager, a dashboard for managing apps, services, orgs, users, and spaces. Until you deploy Apps Manager, you must perform these functions through the of CLI. After Apps Manager has been deployed, Pivotal recommends deselecting the checkbox for this errand on subsequent Elastic Runtime deployments. For more information about the 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 Elastic Runtime. See the [Enabling NFS Volume Services](#) topic for more information.

Step 21: Configure Load Balancers

1. Navigate to the GCP Console and click **Load balancing**.

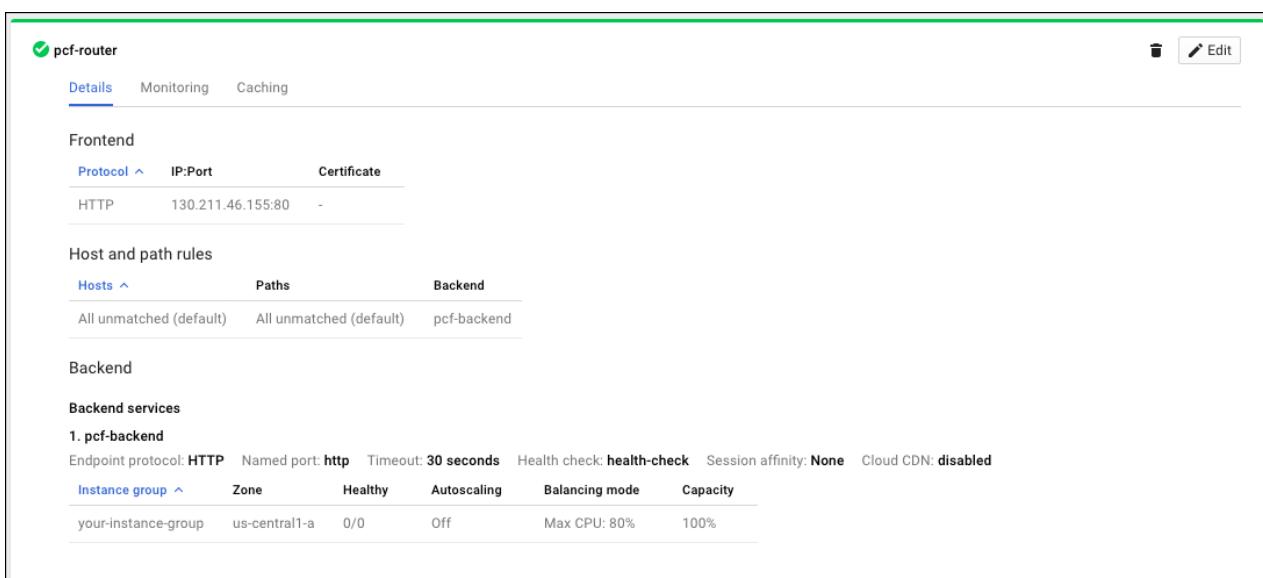


Load balancer	Protocol	
pcf-router	HTTP(S)	 
pcf-ssh	TCP	 
pcf-websockets	TCP	 

To edit load balancing resources like forwarding rules and target proxies, go to the [advanced menu](#).

You should see the SSH load balancer, the HTTP(S) load balancer, the TCP WebSockets load balancer, and optionally, the TCP router that you created in the [Create Load Balancers in GCP](#) section of the *Preparing to Deploy PCF on GCP* topic.

2. Record the name of your SSH load balancer and your TCP WebSockets load balancer. For example, `pcf-ssh` and `pcf-websockets`.
3. Click your HTTP(S) load balancer. For example, `pcf-router`.



Hosts	Paths	Backend
All unmatched (default)	All unmatched (default)	pcf-backend

Instance group	Zone	Healthy	Autoscaling	Balancing mode	Capacity
your-instance-group	us-central1-a	0/0	Off	Max CPU: 80%	100%

4. Under **Backend services**, record the name of the backend service of the HTTP(S) load balancer. For example, `pcf-backend`.

5. In the **Elastic Runtime** tile, click **Resource Config**.

JOB	INSTANCES	PERSISTENT DISK TYPE	VM TYPE	LOAD BALANCERS	INTERNET CONNECTED
Consul	Automatic: 1	Automatic: 1 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)		<input checked="" type="checkbox"/>
NATS	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)		<input checked="" type="checkbox"/>
etcd	Automatic: 1	Automatic: 1 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)		<input checked="" type="checkbox"/>
File Storage	Automatic: 1	Automatic: 100 GB	Automatic: medium.mem (cpu: 1, ram: 1 GB, disk: 8 GB)		<input checked="" type="checkbox"/>
MySQL Proxy	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)		<input checked="" type="checkbox"/>
MySQL Server	Automatic: 1	Automatic: 100 GB	Automatic: large.disk (cpu: 2, ram: 1 GB, disk: 64 GB)		<input checked="" type="checkbox"/>
Backup Prepare Node	0	Automatic: 200 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)		<input checked="" type="checkbox"/>
Apps Manager Database (Postgres)	Automatic: 0	Automatic: 1 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)		<input checked="" type="checkbox"/>
Cloud Controller Database (Postgres)	Automatic: 0	Automatic: 2 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)		<input checked="" type="checkbox"/>
UAA Database (Postgres)	Automatic: 0	Automatic: 10 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)		<input checked="" type="checkbox"/>
UAA	Automatic: 1	None	Automatic: medium.disk (cpu: 2, ram: 4 GB, disk: 32 GB)		<input checked="" type="checkbox"/>
Cloud Controller	Automatic: 1	Automatic: 1 GB	Automatic: medium.disk (cpu: 2, ram: 4 GB, disk: 32 GB)		<input checked="" type="checkbox"/>
HAProxy	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)		<input checked="" type="checkbox"/>
Router	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)	tcp:pcf-websockets,http:pcf-r	<input checked="" type="checkbox"/>
MySQL Monitor	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)		<input checked="" type="checkbox"/>
Clock Global	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)		<input checked="" type="checkbox"/>
Cloud Controller Worker	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)		<input checked="" type="checkbox"/>
Collector	Automatic: 0	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)		<input checked="" type="checkbox"/>
Diego BBS	Automatic: 1	Automatic: 1 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)		<input checked="" type="checkbox"/>
Diego Brain	Automatic: 1	Automatic: 1 GB	Automatic: small (cpu: 1, ram: 2 GB, disk: 8 GB)	tcp:pcf-ssh	<input checked="" type="checkbox"/>
Diego Cell	Automatic: 3	None	Automatic: xlarge.disk (cpu: 1, ram: 16 GB, disk: 128 GB)		<input checked="" type="checkbox"/>
Doppler Server	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)		<input checked="" type="checkbox"/>
Loggregator Trafficcontroller	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)		<input checked="" type="checkbox"/>
TCP Router	0	Automatic: 1 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)		<input checked="" type="checkbox"/>
Push Apps Manager	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)		<input checked="" type="checkbox"/>
Run Smoke Tests	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)		<input checked="" type="checkbox"/>
Push Notifications	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)		<input checked="" type="checkbox"/>
Run Notifications Tests	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)		<input checked="" type="checkbox"/>
Push Notifications UI	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)		<input checked="" type="checkbox"/>
Run Notifications-UI tests	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)		<input checked="" type="checkbox"/>
Push Autoscaling	Automatic: 1	None	Automatic: small (cpu: 1, ram: 2 GB, disk: 8 GB)		<input checked="" type="checkbox"/>
Register Autoscaling Service Broker	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)		<input checked="" type="checkbox"/>
Destroy autoscaling service broker	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)		<input checked="" type="checkbox"/>
Run Autoscaling Tests	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)		<input checked="" type="checkbox"/>
Run CF Acceptance Tests	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)		<input checked="" type="checkbox"/>
Bootstrap	Automatic: 1	None	Automatic: small (cpu: 1, ram: 2 GB, disk: 8 GB)		<input checked="" type="checkbox"/>
Push Pivotal Account	Automatic: 1	None	Automatic: small (cpu: 1, ram: 2 GB, disk: 8 GB)		<input checked="" type="checkbox"/>
MySQL Rejoin Unsafe Errand	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)		<input checked="" type="checkbox"/>

Save

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:pcf-websockets,http:pcf-backend`.



Note: Do not add a space between key/value pairs in the **LOAD BALANCER** field or it will fail.

7. If you have enabled TCP routing in the [Advanced Features](#) 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. Under the **LOAD BALancers** column of the **Diego Brain** row, enter the name of your SSH load balancer prepended with `tcp:` . For example, `tcp:pcf-ssh`.
9. Verify that the **Internet Connected** checkbox for every job is checked to allow the jobs to reach the Internet. This gives all VMs a public IP address that enables outbound Internet access.



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 21: (Optional) Disable Unused Resources

By default, Elastic Runtime uses an internal filestore and internal databases. If you configure Elastic Runtime to use external resources, you can disable the corresponding system-provided resources in Ops Manager to reduce costs and administrative overhead.

For more information regarding scaling instances, see the [Zero Downtime Deployment and Scaling in CF](#) and the [Scaling Instances in Elastic Runtime](#) topics.

Complete the following procedures to disable specific VMs in Ops Manager:

1. Click **Resource Config**.
2. If you configure Elastic Runtime to use an external S3-compatible filestore, edit the following fields:
 - **File Storage**: Enter in **Instances**.
3. If you configure Elastic Runtime to use an external Relational Database Service (RDS), edit the following fields:
 - **MySQL Proxy**: Enter in **Instances**.
 - **MySQL Server**: Enter in **Instances**.
 - **Cloud Controller Database**: Enter in **Instances**.
 - **UAA Database**: Enter in **Instances**.
4. If you are using an External Load Balancer instead of HAProxy, enter in the **Instances** field for **HAProxy**.
5. Click **Save**.

Step 22: Verify and Download Stemcell Version

Verify whether Ops Manager is providing the stemcell version required by Elastic Runtime. If the correct version is already present, you do not need to download a new stemcell.

1. In the Elastic Runtime tile, select **Stemcell**.
2. Verify that the version indicated in the filename matches the version of stemcell required by Elastic Runtime.
 - If Elastic Runtime detects that a stemcell file is present in the Ops Manager Director VM at `/var/tempest/stemcells/`, the Stemcell screen displays filename information.

Stemcell

A stemcell is a template from which Ops Manager creates the VMs needed for a wide variety of components and products.

cf requires BOSH stemcell version 3262 ubuntu-trusty

✓ Using `bosh-stemcell-3262.4-vsphere-esxi-ubuntu-trusty-go_agent.tgz`

Import Stemcell

- If Elastic Runtime cannot detect a stemcell file, the following message displays:

Stemcell

A stemcell is a template from which Ops Manager creates the VMs needed for a wide variety of components and products.

cf requires BOSH stemcell version 3262 ubuntu-trusty

✗ Go to Pivotal Network and download Stemcell 3262.12 ubuntu-trusty.

Import Stemcell

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 Elastic Runtime, or cannot be found by Ops Manager, perform the following steps to download and import a new stemcell file:

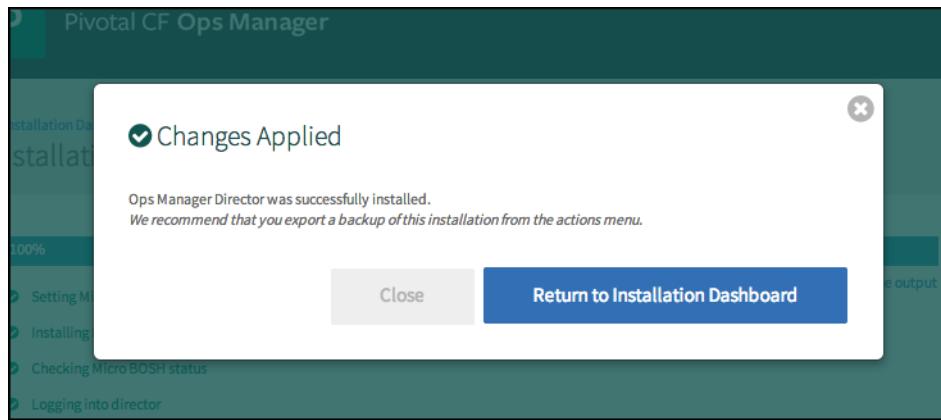
- a. Log in to the [Pivotal Network](#) and click **Stemcells**.
- b. Download the appropriate stemcell version targeted for your IaaS.
- c. In the **Stemcell** section of the Elastic Runtime tile, click **Import Stemcell** to import the downloaded stemcell `.tgz` file.

Step 23: Complete the Elastic Runtime 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.



Deleting a GCP Installation from the Console

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](#) section.
- 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 **External IP addresses**.
8. Select all external IP addresses associated with your PCF deployment, and click **RELEASE STATIC ADDRESS**.
9. 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.
10. Click the upper left icon and select **IAM & Admin**.
11. Click the trashcan icon next to the **bosh** service account you created for PCF and click **REMOVE**.
12. 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 Elastic Runtime Tile Causes Ops Manager Rails Application Crash

Uploading the Elastic Runtime tile causes the Ops Manager Rails application to crash.

Explanation

In compressed format, the ERT tile is 5GB. However, when uncompressed during installation, the ERT 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 [this procedure](#) 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](#).

ERT Deployment Fails - MySQL Monitor replication-canary Job

During installation of the Elastic Runtime 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 Elastic Runtime Router** option in your [Elastic Runtime 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 [Elastic Runtime network configuration](#).

Installing Pivotal Cloud Foundry on OpenStack

Page last updated:

This guide describes how to install [Pivotal Cloud Foundry](#) (PCF) on OpenStack Juno and Kilo distributions.

Supported Versions

Pivotal's automated testing environments have been built using OpenStack releases and distributions based on Havana, Icehouse, Juno, Kilo (Keystone v2, and v3), Liberty, and Mitaka from different vendors including Canonical, EMC, Mirantis, Red Hat, and SUSE. The nature of OpenStack as a collection of interoperable components requires OpenStack expertise to troubleshoot issues that may occur when installing Pivotal Cloud Foundry on particular releases and distributions.

Prerequisites

The following sections describe general requirements for running PCF and specific requirements for running PCF on OpenStack.

General Requirements

The following are general requirements for deploying and managing a PCF deployment with Ops Manager and Elastic Runtime:

- (**Recommended**) Ability to create a wildcard DNS record to point to your router or load balancer. Alternatively, you can use a service such as [xip.io](#). For example, `203.0.113.0.xip.io`.
- (**Recommended**) A network without DHCP available for deploying the Elastic Runtime VMs

 **Note:** If you have DHCP, refer to the Troubleshooting Guide to avoid issues with your installation.

- Sufficient IP allocation:
 - One IP address for each VM instance
 - An additional IP address for each instance that requires static IPs
 - An additional IP address for each errand
 - An additional IP address for each compilation worker: `IPs needed = VM instances + static IPs + errands + compilation workers`
- The most recent version of the [Cloud Foundry Command Line Interface \(cf CLI\)](#)
- One or more NTP servers if not already provided by your IaaS

OpenStack Requirements

To deploy Pivotal Cloud Foundry on OpenStack, you must have a dedicated OpenStack tenant (formerly known as an OpenStack project) that meets the following requirements.

- You must have keystone credentials for the OpenStack tenant, including the following:
 - Auth URL
 - API key
 - Username
 - Project name

- Region
- SSL certificate for your wildcard domain (see below)
- All necessary OpenStack network objects
- The following must be enabled for the tenant:
 - The ability to upload custom images to [Glance](#)
 - The ability to create and modify VM flavors. See the [VM flavor configuration table](#)
 - DHCP
 - The ability to allocate floating IPs
 - The ability for VMs inside a tenant to send messages via the floating IP.
 - Permissions for VMs to boot directly from image
 - One wildcard DNS domain. Pivotal recommends using two wildcard domains if system and apps need to be separated.

 **Note:** For information about how IaaS user roles are configured, refer to the [Pivotal Cloud Foundry IaaS User Role Guidelines](#) topic.

 **Note:** It is possible to avoid using wildcard DNS domains by using a service such as xip.io. However, this option requires granting external internet access from inside VMs.

- Your OpenStack tenant must have the following resources before you install Pivotal Cloud Foundry:
 - 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
 - Neutron networking with floating IP support

 **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.

- Requirements for your Cinder back end:
 - PCF requires RAW root disk images. The Cinder back end for your OpenStack tenant must support RAW.
 - Pivotal recommends that you use a Cinder back end that supports snapshots. This is required for some BOSH functionalities.
 - Pivotal recommends enabling your Cinder back end 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 Elastic Runtime 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.
- 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.

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

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 Elastic Runtime after Deploying PCF on OpenStack](#)

Provisioning the OpenStack Infrastructure

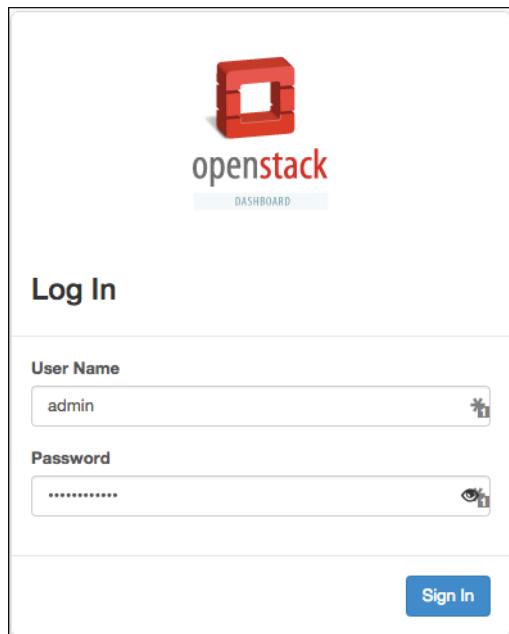
Page last updated:

This guide describes how to provision the OpenStack infrastructure that you need to install Pivotal Cloud Foundry (PCF) OpenStack. This document uses Mirantis Openstack. 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 Elastic Runtime after Deploying PCF on OpenStack](#) topics.

Step 1: Log In to the OpenStack Horizon Dashboard

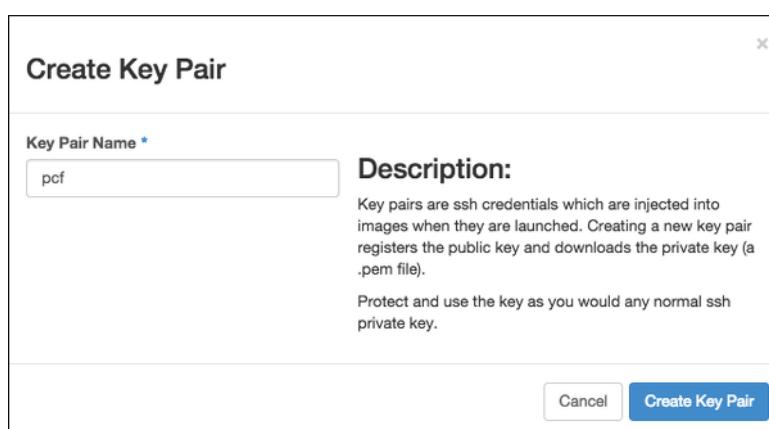
1. Log in to the OpenStack Horizon Dashboard.



The screenshot shows the OpenStack Horizon Dashboard log in page. At the top, there is a logo with the word "openstack" and a "DASHBOARD" button. Below this is a "Log In" section. It contains two input fields: "User Name" with "admin" typed in and a password field with masked text. There is also a "Sign In" button at the bottom of the form.

Step 2: Configure Security

1. In the left navigation of your OpenStack dashboard, click **Project > Compute > Access & Security**.
2. Select the **Key Pairs** tab on the **Access & Security** page.
3. Click **Create Key Pair**.
4. Enter a **Key Pair Name** and click **Create Key Pair**.



The screenshot shows a "Create Key Pair" dialog box. It has a "Key Pair Name" field containing "pcf". To the right, there is a "Description:" section with text about key pairs and a note to protect the private key. At the bottom are "Cancel" and "Create Key Pair" buttons.

5. In the left navigation, click **Access & Security** to refresh the page.
6. Select the **Security Groups** tab. Click **Create Security Group** and create a group with the following properties:
 - **Name:** opsmanager
 - **Description:** Ops Manager

Create Security Group

Name *
opsmanager

Description *
Ops Manager

Description:
Security groups are sets of IP filter rules that are applied to the network settings for the VM. After the security group is created, you can add rules to the security group.

Create Security Group

7. Select the checkbox for the **opsmanager** Security Group and click **Manage Rules**.

<input type="checkbox"/>	Name	Description	Actions
<input type="checkbox"/>	default	default	Manage Rules
<input checked="" type="checkbox"/>	opsmanager	Ops Manager	Manage Rules ▾

Displaying 2 items

8. Add the 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 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

Step 3: Create Ops Manager Image

You can create the Ops Manager image in OpenStack using the OpenStack GUI or using the [Glance CLI](#) client.

Note: If your Horizon Dashboard does not support file uploads, you must use the Glance client.

OpenStack GUI

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 `40`.
 - **Minimum RAM (MB):** Enter `4096`.
 - Deselect the **Public** checkbox.
 - Select the **Protected** checkbox.
4. Click **Create Image**.

Create An Image

Name *
Ops Manager

Description:
Currently only images available via an HTTP URL are supported. The image location must be accessible to the Image Service. Compressed image binaries are supported (.zip and .tar.gz.)
Please note: The Image Location field MUST be a valid and direct URL to the image binary. URLs that redirect or serve error pages will result in unusable images.

Image Source
Image File

Image File [Choose File](#) vm-2015-05-...14fa18e.raw

Format *
Raw

Architecture
[empty]

Minimum Disk (GB) ?
40

Minimum RAM (MB) ?
4096

Public
 Protected

Create Image

Glance CLI

1. Download the **Pivotal Cloud Foundry Ops Manager for OpenStack** image file from [Pivotal Network](#).
2. In a terminal window, run the following command to install the Glance CLI client:

```
$ apt-get install python-glanceclient
```

3. Run `./admin-openrc.sh` to download your `openstack.rc` file and target your OpenStack environment.

```
$ ./admin-openrc.sh
Please enter your OpenStack Password:
```

4. Run the following command to use the Glance CLI client to upload the image file that you downloaded from [Pivotal Network](#):

```
$ glance image-create --progress --disk-format raw --name "Ops Manager" --container-format bare --file PATH/DOWNLOADED-FILE
```

Step 4: Launch Ops Manager VM

1. In the left navigation of your OpenStack dashboard, click **Project > Compute > Images**

2. Click **Launch**.

Images									
<input type="checkbox"/>	Image Name	Type	Status	Public	Protected	Format	Size	Actions	
<input checked="" type="checkbox"/>	Ops Manager	Image	Active	No	No	RAW	3.0 GB	<button>Launch</button> <input type="button" value="▼"/>	
<input type="checkbox"/>	Debian 7.0 (64-bit)	Image	Active	No	No	RAW	3.0 GB	<button>Launch</button> <input type="button" value="▼"/>	

3. Complete the **Details**, **Access & Security**, and **Networking** tabs of the **Launch Instance** form with the information below.

Details Tab

Select the **Details** tab and specify the following details:

- **Availability Zone:** Use the drop-down menu to select an availability zone. You use this availability zone when you [Complete the Availability Zones Pages](#) when [Configuring Ops Manager Director](#).
- **Instance Name:** Enter `Ops Manager`.
- **Flavor:** Select `m1.large`.
- **Instance Count:** Do not change from the default.
- **Instance Boot Source:** Select `Boot from image`.
- **Image Name:** Select the `Ops Manager` image.

Launch Instance

Details *
Access & Security *
Networking *
Post-Creation
Advanced Options

Availability Zone

Instance Name *

Flavor *

Instance Count *

Instance Boot Source *

Image Name

Specify the details for launching an instance.

The chart below shows the resources used by this project in relation to the project's quotas.

Name	m1.large
VCPUs	4
Root Disk	80 GB
Ephemeral Disk	0 GB
Total Disk	80 GB
RAM	8,192 MB

Project Limits

Number of Instances	0 of No Limit Used
Number of VCPUs	0 of No Limit Used
Total RAM	0 of No Limit MB Used

Access & Security Tab

Select the **Access & Security** tab and specify the following details:

- **Key Pair:** Select the key pair that you created in [Step 2: Configure Security](#). You need this key pair to log in to the Ops Manager instance from your workstation.

- **Security Groups:** Select the **opsmanager** checkbox. Deselect all other Security Groups.

Launch Instance

Details * Access & Security * Networking * Post-Creation Advanced Options

Key Pair

Security Groups * default opsmanager

Control access to your instance via key pairs, security groups, and other mechanisms.

Cancel Launch

Networking Tab

1. Select the **Networking** tab.
2. Under **Available networks**, select a private subnet. You add a Floating IP to this network in a later step.
3. Click **Launch**.

Launch Instance

Details * Access & Security * Networking * Post-Creation Advanced Options

Selected networks

NIC:1 net04 (8db1dc3e-c2d0-4425-8949-6994bc09b11a)

Available networks

net04_ext (a353f7f-51bb-4dcf-83aa-eec3332cc864)

Choose network from Available networks to Selected networks by push button or drag and drop, you may change NIC order by drag and drop as well.

Cancel Launch

Step 5: 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 instances shows as *Running*.
3. Record the private **IP Address** of the Ops Manager instance. You use this IP Address when you [Complete the Create Networks Pages](#) in Ops Manager.

<input type="checkbox"/>	Instance Name	Image Name	IP Address	Size	Key Pair	Power State	Actions
<input type="checkbox"/>	██████████2	██████████	██████████0	m1.medium	miran	Running	<input type="button" value="Associate Floating IP"/>
<input type="checkbox"/>	Ops Manager	Ops Manager	192.168.111.54	m1.large	pcf	Running	<input type="button" value="Associate Floating IP"/>

4. Select the **Ops Manager** checkbox. Click the **Actions** drop-down menu and select **Associate Floating IP**.
5. Under IP Address, click **+**.

Manage Floating IP Associations

IP Address * <input type="text" value="Select an IP address"/> +	Select the IP address you wish to associate with the selected instance.
Port to be associated * <input type="text" value="Select a port"/>	
<input type="button" value="Cancel"/> <input type="button" value="Associate"/>	

6. Under **Pool**, select an IP Pool and click **Allocate IP**.

Allocate Floating IP

Pool * <input type="text" value="net04_ext"/>	Description: Allocate a floating IP from a given floating IP pool.
Project Quotas Floating IP (5) 45 Available 	
<input type="button" value="Cancel"/> <input type="button" value="Allocate IP"/>	

7. Under **Port to be associated**, select your **Ops Manager** instance. Click **Associate**.

Manage Floating IP Associations

IP Address * <input type="text" value="216.221.229.32"/> +	Select the IP address you wish to associate with the selected instance.
Port to be associated * <input type="text" value="Ops Manager: 192.168.111.54"/>	
<input type="button" value="Cancel"/> <input type="button" value="Associate"/>	

Step 6: 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:** Select `private`.

Create Container

Container Name *

Container Access *

Description:

A container is a storage compartment for your data and provides a way for you to organize your data. You can think of a container as a folder in Windows ® or a directory in UNIX ®. The primary difference between a container and these other file system concepts is that containers cannot be nested. You can, however, create an unlimited number of containers within your account. Data must be stored in a container so you must have at least one container defined in your account prior to uploading data.

Note: A Public Container will allow anyone with the Public URL to gain access to your objects in the container.

Step 7: 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 IP address that you used for Ops Manager. You must use this fully qualified domain name when you log into Ops Manager in the Configure Ops Manager Director for OpenStack step below.

Step 8: 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 Elastic Runtime 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](#).

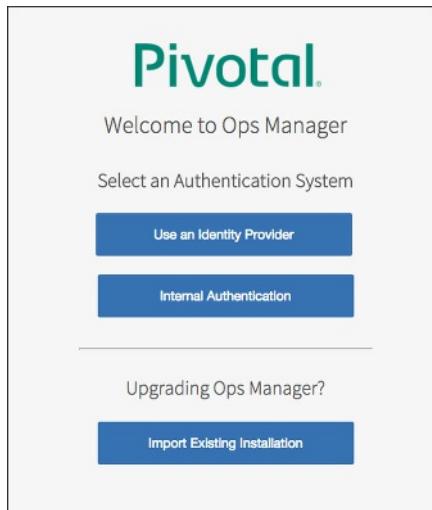
Before beginning this procedure, ensure that you have successfully completed all steps in the [Provisioning the OpenStack Infrastructure](#) topic. After you complete this procedure, follow the instructions in the [Installing Elastic Runtime after Deploying PCF on OpenStack](#) topic.

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 are using 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**.

The screenshot shows the 'Internal Authentication' setup page. It includes fields for 'Username', 'Password', 'Password confirmation', 'Decryption passphrase', 'Decryption passphrase confirmation', and proxy settings ('Http proxy', 'Https proxy', 'No proxy'). There is also a checkbox for agreeing to the terms and conditions, which links to the 'End User License Agreement'. A 'Setup Authentication' button is at the bottom.

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

The screenshot shows the sign-in page with a 'Welcome!' message. It has fields for 'Email' and 'Password', and a 'SIGN IN' button. A horizontal line is present below the sign-in form.

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.

Access & Security

Security Groups Key Pairs Floating IPs API Access

[Download OpenStack RC File](#) [Download EC2 Metadata](#)

API Endpoints

Service	Service Endpoint
Compute	http://2.165.150.122:8774/v2.1
Network	http://2.165.150.122:8773/v2.0
Volumev2	http://2.165.150.122:8776/v2.0
S3	http://2.165.150.122:80
Image	http://2.165.150.122:80
Cloudformation	http://2.165.150.122:80/v1/
Volume	http://2.165.150.122:8772
EC2	http://2.165.150.122:8773/services/Cloud
Orchestration	http://2.165.150.122:8774/v2.0
Object Store	http://2.165.150.122:80/v1/AUTH_6f7007250c7200140a4d0a2000000000
Identity	http://2.165.150.122:5000/v2.0

Displaying 11 items

3. In the PCF Ops Manager Installation Dashboard, click the **Ops Manager Director** tile.



4. Select **OpenStack Config**.

[Installation Dashboard](#)

Ops Manager Director

Settings Status Credentials

Openstack Config Advanced Infrastructure Config Director Config Create Availability Zones Create Networks Assign AZs and Networks Security Resource Config

Openstack Management Console Config

Authentication URL*
 URL to the Openstack Identity Endpoint

Keystone Version
 v2 v3

Domain

Username*

OpenStack Management Console Config

Configure the OpenStack Management Console settings.

Required Fields

Fields marked with an asterisk (*) are required.

OpenStack Management Console Settings

Identity

Authentication URL: [Identity Service Endpoint](#)

Keystone Version: v3

Domain: [Default Domain](#)

Username: opsmanager

Password: [Change](#)

Tenant: corgi

Region: RegionOne

Ignore Server Availability Zone

Security Group Name: corgi

Key Pair Name*: corgi_key

SSH Private Key*: [Change](#)

API SSL Certificate*: [Change](#)

-----BEGIN CERTIFICATE-----
MI[REDACTED]
-----END CERTIFICATE-----

Disable DHCP

Save

5. Complete the **OpenStack Management Console Config** page with the following information:

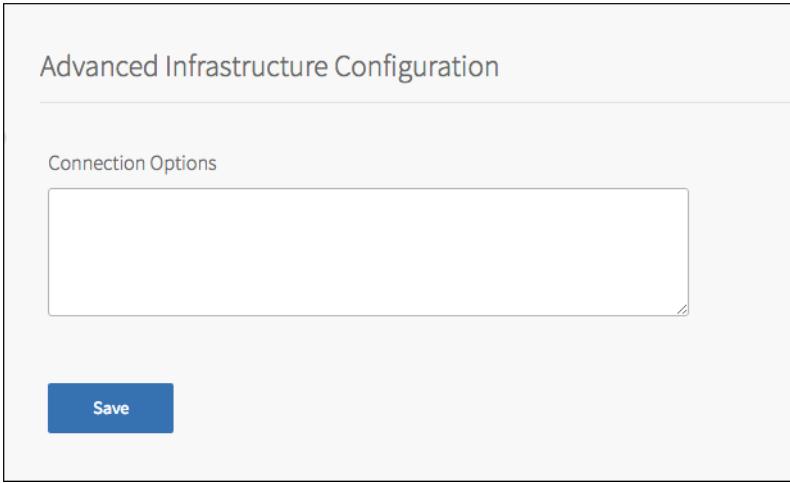
- **Authentication URL:** Enter the Service Endpoint for the Identity service that you recorded in a previous step.
- **Keystone Version:** Choose a Keystone version. If you choose **v3**, you must enter a **Domain** to authenticate against.
- **Username:** Enter your OpenStack Horizon username.
- **Password:** Enter your OpenStack Horizon password.
- **Tenant:** Enter your OpenStack tenant name.
- **Region:** Enter `RegionOne`, or another region if recommended by your OpenStack administrator.
- **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.

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**.

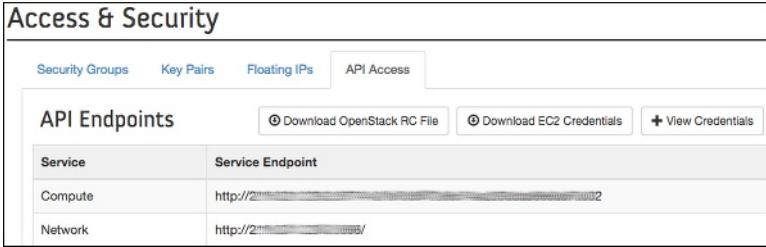


The screenshot shows the 'Advanced Infrastructure Configuration' page. At the top, there's a title bar. Below it, a section titled 'Connection Options' contains a large empty text area. At the bottom right of this section is a blue 'Save' button.

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 the left navigation of your OpenStack dashboard, click **Project > Compute > Access & Security**. Select the **API Access** tab.



The screenshot shows the 'Access & Security' section with the 'API Access' tab selected. Under the 'API Endpoints' heading, there are two entries: 'Compute' with endpoint `http://2.11.11.11:8774/v2.0` and 'Network' with endpoint `http://2.11.11.11:8773`. Below each entry are three buttons: 'Download OpenStack RC File', 'Download EC2 Credentials', and '+ View Credentials'.

2. Click **Download EC2 Credentials**.
3. Unzip the downloaded credentials. In a text editor, open the `ec2rc.sh` file. Depending on your configuration, you may use the contents of this file to complete the Ops Manager **Director Config** page.
4. In Ops Manager, select **Director Config**.

Director Config

NTP Servers (comma delimited)*

Metrics IP Address

Enable VM Resurrector Plugin

Enable Post Deploy Scripts

Recreate all VMs
This will force BOSH to recreate all VMs on the next deploy. Persistent disk will be preserved.

Enable bosh deploy retries
This will attempt to re-deploy a failed deployment up to 5 times.

Keep Unreachable Director VMs

5. Enter one or more NTP servers in the **NTP Servers (comma delimited)** field. For example, `us.pool.ntp.org`.
6. (Optional) Enter your **Metrics IP Address** if you are [Using JMX Bridge](#).
7. Select the **Enable VM Resurrector Plugin** checkbox to enable the Ops Manager Resurrector functionality and increase Elastic Runtime availability.
8. 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.
9. Select **Recreate all VMs** to force BOSH to recreate all VMs on the next deploy. This process does not destroy any persistent disk data.
10. Select **Enable bosh deploy retries** if you want Ops Manager to retry failed BOSH operations up to five times.
11. Select **Keep Unreachable Director VMs** if you want to preserve Ops Manager Director VMs after a failed deployment for troubleshooting purposes.
12. Select **HM Pager Duty Plugin** to enable Health Monitor integration with PagerDuty.

HM Pager Duty Plugin

Service Key*

HTTP Proxy

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

HM Email Plugin

Host*
smtp.example.com

Port*
25

Domain*
cloudfoundry.example.com

From*
user2@example.com

Recipients*
user@example.com, user1@example.com

Username
user

Password
.....

Enable TLS

13. 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.

14. For **Blobstore Location**, select **S3 Compatible Blobstore** and complete the following steps using information from the `ec2rc.sh` file:

Blobstore Location

Internal

S3 Compatible Blobstore

S3 Endpoint*

Bucket Name*

XXXXXXXXXX

Access Key*

Secret Key*

[Change](#)

V2 Signature

V4 Signature

Region*

- **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.

15. 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.

Database Location

Internal

External MySQL Database

Host*

Port*

3306

Username*

Password*

.....

[Change](#)

Database*

bosh

Max Threads

Director Hostname

opsdirector.example.com

Save

16. For **Max Threads**, enter the number of operations the Ops Manager Director can perform simultaneously.
17. (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](#).
18. 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**.

Create Availability Zones

Availability Zones

▼ nova

Openstack Availability Zone*

nova

Save

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

- In the left navigation of your OpenStack dashboard, click **Project > Network > Networks**

Networks						
<input type="checkbox"/>	Name	Subnets Associated	Shared	Status	Admin State	Actions
<input type="checkbox"/>	net04_ext	net04_ext_subnet 255.255.255.0/24	No	ACTIVE	UP	<button>Edit Network</button>
<input type="checkbox"/>	net04	net04_subnet 192.168.111.0/24	No	ACTIVE	UP	<button>Edit Network</button>

- 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.

Network Overview

Name
net04

ID
8db1dc3e-XXXXXXXXXXb11a

Project ID
feXXXXXXXXXX2

Status
ACTIVE

Admin State
UP

Shared
No

External Network
No

Provider Network
Network Type: vlan
Physical Network: physnet2
Segmentation ID: 1003

Subnets

<input type="checkbox"/>	Name	Network Address	IP Version	Gateway IP	Actions
<input type="checkbox"/>	net04_subnet	192.168.111.0/24	IPv4	192.168.111.1	<button>Edit Subnet</button>

Displaying 1 item

Ports

Name	Fixed IPs	Attached Device	Status	Admin State	Actions
(3906d7ec)	192.168.111.2	network:dhcp	ACTIVE	UP	<button>Edit Port</button>
(472a6c20)	192.168.111.1	network:router_interface	ACTIVE	UP	<button>Edit Port</button>
(67904dcb)	192.168.111.10	compute:None	ACTIVE	UP	<button>Edit Port</button>
(70b7af50)	192.168.111.54	compute:nova	ACTIVE	UP	<button>Edit Port</button>
(7232bd69)	192.168.111.3	network:dhcp	ACTIVE	UP	<button>Edit Port</button>

- In Ops Manager, select **Create Networks**.

Create Networks

Warning: Pivotal recommends keeping the IP settings throughout the life of your installation. Ops Manager may prevent you from changing them in the future. Contact Pivotal support for help completing such a change.

Verification Settings

Enable ICMP checks

Networks

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

first-network

Name* A unique name for this network

Subnets

Network ID*

CIDR*

Reserved IP Ranges

DNS*

Gateway*

Availability Zones* nova

second-network

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.



Note: To use the [Single Sign-On for PCF](#) service, you must configure a network with only one subnet.

- 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**.

Assign AZs and Networks

The Ops Manager Director is a single instance.

Choose the availability zone in which to place that instance. It is highly recommended that you backup this VM on a regular basis to preserve settings.

Singleton Availability Zone

nova

Network

first-network

Save

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

Security

Trusted Certificates

-----BEGIN CERTIFICATE-----
TH...
-----END CERTIFICATE-----

These certificates enable BOSH-deployed components to trust a custom root certificate.

Generate VM passwords or use single password for all VMs

Generate passwords
 Use default BOSH password

Save

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 Trusted Registries for running app instances in Docker containers, use this field to enter your certificate for your private Docker Trusted Registry. See the [Using Docker Trusted 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 Resource Config Page

1. Select **Resource Config**.

Resource Config

JOB	INSTANCES	PERSISTENT DISK TYPE	VM TYPE
Ops Manager Director	Automatic: 1	Automatic: 50 GB	Automatic: medium.disk (cpu: 2, ram: 4 GB)
Master Compilation Job	Automatic: 4	None	Automatic: large.cpu (cpu: 4, ram: 4 GB, di: 4)

Save

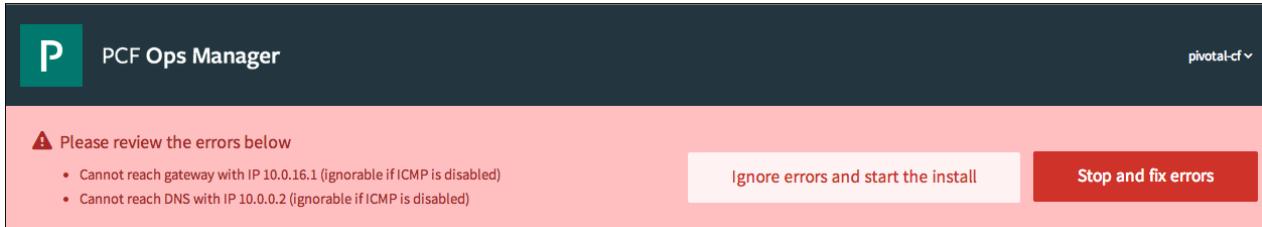
2. Adjust any values as necessary for your deployment, such as increasing the persistent disk size. Select **Automatic** from the drop-down menu to provision the amount of persistent disk predefined by the job. If the persistent disk field reads **None**, the job does not require persistent disk space.

 **Note:** 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.

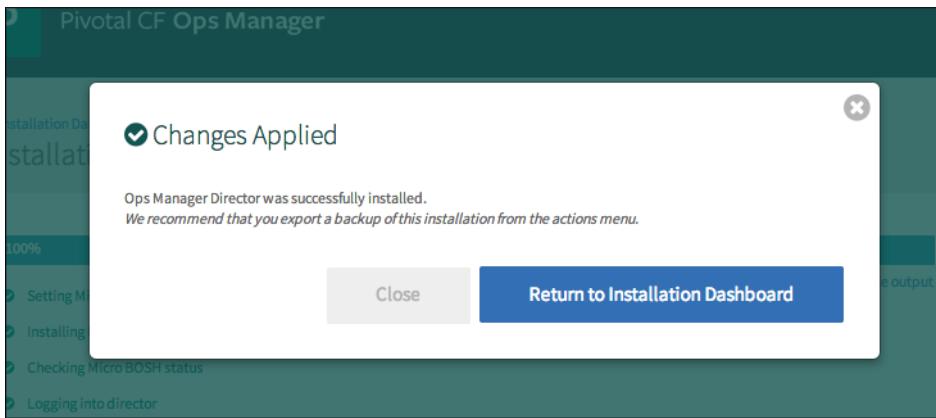
3. Click **Save**.

Step 10: Complete 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 Elastic Runtime after Deploying PCF on OpenStack](#) topic.

Return to [Installing Pivotal Cloud Foundry on OpenStack](#).

Installing Elastic Runtime after Deploying Pivotal Cloud Foundry on OpenStack

Page last updated:

This topic describes how to install and configure Elastic Runtime after deploying [Pivotal Cloud Foundry](#) (PCF) on OpenStack.

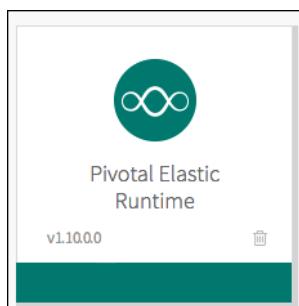
Use this topic when [Installing Pivotal Cloud Foundry on OpenStack](#).

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.

 **Note:** If you are performing an upgrade to PCF 1.8, please review [Upgrading Pivotal Cloud Foundry](#) for critical upgrade information.

Step 1: Add Elastic Runtime to Ops Manager

1. Navigate to the Pivotal Cloud Foundry Operations Manager Installation Dashboard.
2. Click the Pivotal Network link on the left to add Elastic Runtime 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 Elastic Runtime.

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 Elastic Runtime.

[Installation Dashboard](#)

Pivotal Elastic Runtime

Settings Status Credentials Logs

Assign AZs and Networks

AZ and Network Assignments

Place singleton jobs in first-az

Balance other jobs in first-az

Network

Save

- Domains
- Networking
- Application Containers
- Application Developer Controls
- Application Security Groups
- Authentication and Enterprise SSO
- Databases

- Click **Save**.

Note: When you save this form, a verification error displays because the PCF security group blocks ICMP. You can ignore this error.

PCF Ops Manager

⚠ Please review the errors below

- Cannot reach gateway with IP 10.0.16.1 (ignorable if ICMP is disabled)
- Cannot reach DNS with IP 10.0.0.2 (ignorable if ICMP is disabled)

All errors will be reverified before installation.

Step 3: Configure Domains

- Select **Domains**.

Elastic Runtime hosts applications at subdomains under its apps domain and assigns system components to subdomains under its system domain. You need to configure a wildcard DNS for both the apps domain and system domain. The two domains can be the same, although this is not recommended.

System Domain *

Apps Domain *

Save

2. Enter the system and application domains.
 - The **System Domain** defines your target when you push apps to Elastic Runtime.
 - The **Apps Domain** defines where Elastic Runtime 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 your own load balancer or the HAProxy load balancer. Find your load balancer type in 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 your subnet.

LOAD BALANCER	ROUTER IP FIELD VALUE	HAProxy IP FIELD VALUE
Your own load balancer	Enter the IP address or addresses for PCF that you registered with your load balancer. Refer to the Using Your Own Load Balancer topic for help using your own load balancer with PCF.	Leave this field blank.
HAProxy load balancer	Leave this field blank.	Enter at least one HAProxy IP address. Point your DNS to this address. For more information, see Configuring SSL/TLS Termination at HAProxy .

For help understanding the Elastic Runtime architecture, refer to the [Architecture](#) topic.

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.

Configure security and routing services for your platform. It is usually preferable to use your own load balancer instead of an HAProxy instance as your point-of-entry to the platform.

Router IPs

SSH Proxy IPs

HAProxy IPs

TCP Router IPs

5. Under **Configure the point-of-entry to this environment** choose one of the following options:

 **Note:** For details about the different SSL/TLS termination point options, how they correspond to different points-of-entry for Elastic Runtime, and related certificate requirements, see the [Providing a Certificate for your SSL Termination Point](#) topic.

- **Forward SSL to Elastic Runtime Router:** Select this option if your deployment uses an external load balancer that can forward encrypted traffic to the Elastic Runtime Router, or for a development environment that does not require load balancing. Complete the fields for the **Router SSL Termination Certificate and Private Key** and **Router SSL Ciphers**.

Select one of the following point-of-entry options:^{*}

Forward SSL to Elastic Runtime Router. Assumes an external load balancer is configured to forward encrypted traffic.

Router SSL Termination Certificate and Private Key *

```
-----BEGIN CERTIFICATE-----
[REDACTED]
-----END CERTIFICATE-----
```

Change
Required for SSL encryption between the load balancer and router(s).

Router SSL Ciphers

Forward unencrypted traffic to Elastic Runtime Router. Assumes an external load balancer is configured to forward unencrypted traffic.
 Forward SSL to HA Proxy. Like first option - Assumes an external load balancer is configured to forward encrypted traffic.

- **Forward unencrypted traffic to Elastic Runtime Router:** Select this option if your deployment uses an external load balancer that cannot forward encrypted traffic to the Elastic Runtime Router, or for a development environment that does not require load balancing.

Select one of the following point-of-entry options:^{*}

Forward SSL to Elastic Runtime Router. Assumes an external load balancer is configured to forward encrypted traffic.
 Forward unencrypted traffic to Elastic Runtime Router. Assumes an external load balancer is configured to forward unencrypted traffic.
 Forward SSL to HA Proxy. Like first option - Assumes an external load balancer is configured to forward encrypted traffic.

Disable SSL certificate verification for this environment
 Disable insecure cookies on the Router

- **Forward SSL to HAProxy:** Select this option to use HAProxy as your first point of entry. Complete the fields for **SSL Certificate and Private Key**, and **HAProxy SSL Ciphers**. Select **Disable HTTP traffic to HAProxy** if you want the HAProxy to only allow HTTPS

traffic.

6. 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.

7. Select the **Disable insecure cookies on the Router** checkbox to set the secure flag for cookies generated by the router.
 8. 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](#).

- Disable SSL certificate verification for this environment
- Disable insecure cookies on the Router
- Enable Zipkin tracing headers on the router

- 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.
 - The **Loggregator Port** defaults to `443` if left blank. Enter a new value to override the default.
 - (Optional) Use the **Applications Subnet** field if you need to avoid address collision with a third-party service on the same subnet as your apps. Enter a CIDR subnet mask specifying the range of available IP addresses assigned to your app containers. The IP range must be

different from the network used by the system VMs.

12. (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.

Choose whether to enable route services. Route services enable you to proxy requests to your app over TLS to arbitrary URLs before hitting your app.*

Enable route services
 Disable route services

Loggregator Port

Applications Subnet (Only change this if you need to avoid address collision with a third-party service on the same subnet.) *

`10.254.0.0/22`

Applications Network Maximum Transmission Unit (MTU) (in bytes) *

`1454`

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

14. (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.

15. (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.

Router Timeout to Backends (in seconds) (min: 1) *

`900`

Load Balancer Unhealthy Threshold *

`0`

Load Balancer Healthy Threshold *

`20`

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

Router Max Idle Keepalive Connections (min: 0, max: 50000) *

`0`

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

- Select the **Enable TCP Routing** radio button.
- 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.

Enable TCP requests to your apps via specific ports on the TCP router. You will want to configure a load balancer to forward these TCP requests to the TCP routers. If you do not have a load balancer, then you can also send traffic directly to the TCP router.*

- Select this option if you prefer to enable TCP Routing at a later time
- Enable TCP Routing

TCP Routing Ports (one-time configuration, if you want to update this value you can via the CF CLI) *

1024-1123

- 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. To disable TCP routing, select the **Select this option if you prefer to enable TCP Routing at a later time** radio button. For more information, see the [Configuring TCP Routing in Elastic Runtime](#) topic.

18. Click **Save**.

Step 5: Configure Application Containers

1. Select **Application Containers**.

Enable microservice frameworks, private Docker registries, and other services that support your applications at a container level.

Enable Custom Buildpacks

Allow SSH access to app containers

Enable SSH when an app is created

Private Docker Insecure Registry Whitelist

10.10.10.10:8888,example.com:8888



Docker Images Disk-Cleanup Scheduling on Cell VMs*

- Never clean up Cell disk-space
- Routinely clean up Cell disk-space
- Clean up disk-space once threshold is reached

Threshold of Disk-Used (MB) (min: 1) *

Max Inflight Container Starts *

Save

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 Elastic Runtime 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 Trusted Registries](#) topic for more information.
6. Select your preference for **Docker Images Disk-Cleanup on Cell VMs**. If you choose **Clean up disk-space once threshold is reached** enter a **Threshold of Disk-Used** in megabytes.
7. Optionally, 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. Click **Save**.

Step 6: Configure Application Developer Controls

1. Select **Application Developer Controls**.

Configure restrictions and default settings for applications pushed to Elastic Runtime.

Maximum File Upload Size (MB) (min: 1024, max: 2048) *

Default App Memory (MB) (min: 64, max: 2048) *

Default App Memory Quota per Org (min: 10240, max: 102400) *

Maximum Disk Quota per App (MB) (min: 512, max: 10240) *

Default Disk Quota per App (MB) (min: 512, max: 10240) *

Default Service Instances Quota per Org (min: 0, max: 1000) *

Save

2. Enter your intended maximum file upload size.
3. Enter your default RAM memory allocation per app.
4. Enter your default total RAM memory (RAM) quota per Org. You can change this in the CLI.
5. Enter your maximum and default disk quotas per app.
6. Enter your default service instances quota per Org. You can change this in the CLI.
7. Click **Save**.

Step 7: Review Application Security Groups

Setting appropriate [Application Security Groups](#) is critical for a secure deployment. Type in the box to acknowledge that once the Elastic Runtime deployment completes, you will review and set the appropriate application security groups.

Setting appropriate Application Security Groups that control application network policy is the responsibility of the Elastic Runtime administration team. Please refer to the Application Security Groups topic in the Pivotal Cloud Foundry documentation for more detail on completing this activity after the Elastic Runtime deployment completes.

Type X to acknowledge that you understand this message *

Save

Step 8: Configure Authentication and Enterprise SSO

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

Configure your user store access, which can be an internal user store (managed by Cloud Foundry's UAA) or an external user store (LDAP or SAML). You can also adjust the lifetimes of authentication tokens.

Configure your UAA user account store with either internal or external authentication mechanisms*

- Internal UAA (provided by Elastic Runtime; configure your password policy below)

Minimum Password Length *

Minimum Uppercase Characters Required for Password *

Minimum Lowercase Characters Required for Password *

Minimum Numerical Digits Required for Password *

Minimum Special Characters Required for Password *

Number of Months Before Password Expires *

Maximum Password Entry Attempts Allowed *

- SAML Identity Provider

- LDAP Server

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 Elastic Runtime* 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 Elastic Runtime* topic.

- (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.
- (Optional) Customize the text prompts used for username and password from the cf CLI and Apps Manager login popup.
- (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 public IP address(es), append a regular expression or regular expressions to match the public IP address(es).

Apps Manager Access Token Lifetime (in seconds) *

Apps Manager Refresh Token Lifetime (in seconds) *

Cloud Foundry CLI Access Token Lifetime (in seconds) *

Cloud Foundry CLI Refresh Token Lifetime (in seconds) *

Set the lifetime of the refresh token for the Cloud Foundry CLI.

Customize Username Label (on login page) *

Customize Password Label (on login page) *

Proxy IPs Regular Expression *

Save

- Click **Save**.

Step 9: Configure System Databases

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

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:

- Select **Databases**.

Place the databases used by Elastic Runtime components like Cloud Controller and UAA.

Choose the location of your system databases*

- Internal Databases - MySQL and Postgres (the Postgres DBs are not highly-available, but this selection is required if you want to keep your system data from a pre-1.6.0 Elastic Runtime that you upgraded)
- Internal Databases - MySQL (preferred for complete high-availability)
- External Databases (preferred if, for example, you use AWS RDS)

Save

2. Select **Internal Databases - MySQL**.

3. Click **Save**.

Then proceed to [Step 10: \(Optional\) Configure Internal MySQL](#) 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. You can configure a different database provider that provides MySQL support, such as Google Cloud SQL.

To create your Elastic Runtime databases, perform the following steps:

- 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
```

- SSH in to your Ops Manager using the [Ops Manager FQDN](#) and the username `ubuntu`:

```
ssh ubuntu@OPS_MANAGER_FQDN
```

- 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=RDHOSTNAME --user=RDSUSERNAME --password=RDPASSWORD
```

- Run the following MySQL commands to create databases for the seven Elastic Runtime components that require a relational database:

```
CREATE database uaa;
CREATE database cedb;
CREATE database notifications;
CREATE database autoscale;
CREATE database app_usage_service;
CREATE database routing;
CREATE database diego;
CREATE database account;
CREATE database nfsvolume;
```

- Type `exit` to quit the MySQL client, and `exit` again to close your connection to the Ops Manager VM.
- In Elastic Runtime, select **Databases**.
- Select the **External Databases** option.

Place the databases used by Elastic Runtime components like Cloud Controller and UAA.

Choose the location of your system databases*

- Internal Databases - MySQL and Postgres (the Postgres DBs are not highly-available, but this selection is required if you want to keep your system data from a pre-1.6.0 Elastic Runtime that you upgraded)
- Internal Databases - MySQL (preferred for complete high-availability)
- External Databases (preferred if, for example, you use AWS RDS)

Hostname *

TCP Port *

App Usage Service database username *

App Usage Service database password *

8. Complete the following fields in Elastic Runtime:

Elastic Runtime Field	Notes
Hostname	Specify the hostname of the database server.
TCP Port	Specify the port of the database server.
DATABASE-NAME database username	Specify a unique username that can access this specific database on the database server.
DATABASE-NAME database password	Specify a password for the provided username.

9. Click **Save**.

Step 10: (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 [Proxy](#) section of the *MySQL for PCF* topic for more information.

Only configure this section if you selected Internal Databases - MySQL or Internal Databases - MySQL and Postgres in the previous Databases section.

A proxy tier routes MySQL connections from internal components to healthy cluster nodes. Configure DNS and/or your own load balancer to point to multiple proxy instances for increased availability. TCP healthchecks can be configured against port 1936.

The automated backups functionality works with any S3-compatible file store that can receive your backup files.

MySQL Proxy IPs

MySQL Service Hostname

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.
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.

Replication canary time period *
<input type="text" value="30"/>
Replication canary read delay *
<input type="text" value="20"/>
E-mail address (required) *
<input type="text" value="email@example.com"/>

7. Under **Automated Backups Configuration**, choose one of three options for MySQL backups:
 - **Disable automatic backups of MySQL**
 - **Enable automated backups from MySQL to an S3 bucket or other S3-compatible file stores** saves your backups to an existing Amazon Web Services (AWS) or [Ceph](#) S3-compatible blobstore.

Automated Backups Configuration*

Disable automated backups of MySQL

Enable automated backups from MySQL to an S3 bucket or other S3-compatible file store

S3 Endpoint URL *

S3 Bucket Name *

Bucket Path *

AWS Access Key ID *

AWS Secret Access Key *

Cron Schedule *

Backup All Nodes

Enable automated backups from MySQL to a remote host via SCP

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 **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 automated backups from MySQL to a remote host via SCP** saves your backups to a remote host using secure copy

Automated Backups Configuration*

Disable automated backups of MySQL

Enable automated backups from MySQL to an S3 bucket or other S3-compatible file store

Enable automated backups from MySQL to a remote host via SCP

Hostname *

Port *

Username *

Private key *

Destination directory *

Cron Schedule *

Backup All Nodes

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 Elastic Runtime tile to 0.

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

- 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.

Server Activity Logging*

Disable server activity logging
 Enable server activity logging

Event types *

connect,query

Load Balancer Healthy Threshold *

0

Load Balancer Unhealthy Threshold *

0

Save

9. 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.

10. Click **Save**.

Step 11: 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 Elastic Runtime 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 Elastic Runtime file storage, perform the following steps:

1. In the Elastic Runtime tile, select **File Storage**.

This section determines where you would like to place your Elastic Runtime Cloud Controller's file storage.

Configure your Cloud Controller's filesystem*

- Internal WebDAV (provided by Elastic Runtime)
- External S3-Compatible File Store (if you want to use a service like S3 or Ceph)

URL Endpoint *

`https://s3.amazonaws.com`

Access Key *

XYZ1234567

Secret Key *

.....

S3 Signature Version*

V4 Signature

Region*

US West (N. California)

- Server-side Encryption
(available for AWS S3 only)

Buildpacks Bucket Name *

PcfElasticRuntimeS3Buildpack

Droplets Bucket Name *

PcfElasticRuntimeS3DropletBu

Packages Bucket Name *

PcfElasticRuntimeS3PackagesB

Resources Bucket Name *

PcfElasticRuntimeS3Resources

- External Google Cloud Storage
- External Azure Storage

Save

2. Select the **External S3-Compatible Filestore** option and complete the following fields:

- Enter the **URL Endpoint** for your filestore.
- Enter your **Access Key** and **Secret Key**.
- 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.
- Enter values for the remaining fields as follows:

Ops Manager Field	Description
-------------------	-------------

Buildpacks Bucket Name	S3 bucket for storing app buildpacks.
Droplets Bucket Name	S3 bucket for storing 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	S3 bucket for storing 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	S3 bucket for storing app resources. Pivotal recommends that you use a unique bucket name for app resources, but you can also use the same name as above.

- Click **Save**.

 **Note:** For more information regarding AWS S3 Signatures, see the [Authenticating Requests](#) 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 12: (Optional) Configure System Logging

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

- Select **System Logging**.

Configure system logging. Complete the External Syslog fields only if using an external syslogd server.

Enable Cloud Controller security event logging

External Syslog Aggregator Hostname
 

External Syslog Aggregator Port
 The typical syslogd port is 514. Ensure syslogd is listening on external interfaces.

External Syslog Network Protocol

Syslog Drain Buffer Size (# of messages) *

Save

- 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, and request result, in the Common Event Format (CEF).
- 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 Elastic Runtime network, accept TCP connections, and use the RELP protocol. Ensure your syslog server listens on external interfaces.

- Select an **External Syslog Network Protocol** to use when forwarding logs.
- 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**.

Step 13: (Optional) Customize Apps Manager

The **Custom Branding** and **Apps Manager** sections customize the appearance and functionality of Apps Manager. Refer to [Custom Branding](#) and [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.

Customize colors, images, and text for Apps Manager and the Cloud Foundry login portal.

Company Name

Accent Color

Main Logo (PNGs only)

Square Logo/Favicon (PNGs only)

Footer Text
 Defaults to 'Pivotal Software Inc. All rights reserved.'

Footer Links
You may configure up to three links in the Apps Manager footer Add

Classification Header/Footer Background Color

Classification Header/Footer Text Color

Classification Header Content

Classification Footer Content

Save

2. Click **Save** to save your settings in this section.
3. Select **Apps Manager**. Use this section to configure page names and sidebar links in the **Apps Manager** and **Marketplace** pages.

Configure Apps Manager

Display Marketplace Service Plan Prices

Supported currencies as json *****

 Define the currency codes and associated symbols (defaults to "{ \"usd\": \"\$\", \"eur\": \"€\" }" if blank)

Product Name

Marketplace Name

Customize Sidebar Links

You may configure up to 10 links in the Apps Manager sidebar

Add 

▶ Marketplace	
▶ Docs	
▶ Tools	

Save

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

Step 14: (Optional) Configure Email Notifications

Elastic Runtime 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**.

Configure Simple Mail Transfer Protocol for the Notifications application to send email notifications about your deployment. This application is deployed as an errand in Elastic Runtime. If you do not need this service, leave this section blank and disable the Notifications and Notifications UI errands.

From Email

Address of SMTP Server

Port of SMTP Server

SMTP Server Credentials

[Change](#)

SMTP Enable Automatic STARTTLS

SMTP Authentication Mechanism*

SMTP CRAMMD5 secret

Save

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 15: (Optional) Add CCDB Restore Key

Perform this step if all of the following are true:

- You deployed Elastic Runtime previously.
- You then stopped Elastic Runtime or it crashed.
- You are re-deploying Elastic Runtime with a backup of your Cloud Controller database.

1. Click **Restore CCDB Encryption Key**.
2. Enter your **Cloud Controller DB Encryption Key**.

You must enter your original database encryption key here once you have deployed Elastic Runtime for the first time. Anytime thereafter your encryption key will remain the same, if you specify the same key here. This is important for backup and restore of data as well as upgrades.

Cloud Controller DB Encryption Key

...

Leave blank unless restoring a Cloud Controller Database exported from a previous installation.

Save

See [Backing Up Pivotal Cloud Foundry](#) for more information.

Step 16: Configure Smoke Tests

The Smoke Tests errand runs basic functionality tests against your Elastic Runtime 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 org and space**, which creates an ad-hoc org and space for running smoke tests and deletes them afterwards. Otherwise, select **Specified org and space** and complete the fields to specify where you want to run smoke tests.

Specify a Cloud Foundry organization and space where smoke tests can run if in the future you delete your Elastic Runtime deployment domains.

Choose whether to create a new org and space on demand for smoke tests*

- Temporary org and space (This org and space are deleted after smoke tests finish.)
 Specified org and space (The org and space must have a domain available for routing.)

Organization *

Space *

Domain *

Save

3. Click **Save**.

Step 17: (Optional) Enable Advanced Features

The Advanced Features section of Elastic Runtime 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

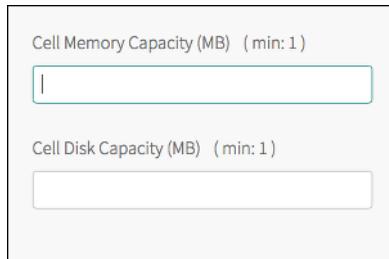
If your apps do not use the full allocation of disk space and memory set in the **Resource Config** tab, you might want use this feature. These fields control the amount to overcommit disk and memory resources to each Diego Cell VM.

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**.



The screenshot shows a form with two input fields. The first field is labeled "Cell Memory Capacity (MB) (min: 1)" with a text input box containing a single vertical bar character. The second field is labeled "Cell Disk Capacity (MB) (min: 1)" with an empty text input box.

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

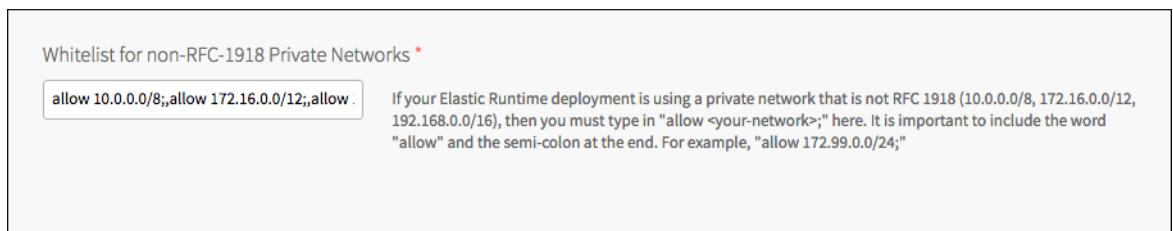
Some private networks require extra configuration so that internal file storage (WebDAV) can communicate with other PCF processes.

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.



The screenshot shows a text input field for the whitelist. The placeholder text is "Whitelist for non-RFC-1918 Private Networks *". Below the input field is a note: "allow 10.0.0.0/8;allow 172.16.0.0/12;allow . If your Elastic Runtime deployment is using a private network that is not RFC 1918 (10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16), then you must type in "allow <your-network>;" here. It is important to include the word "allow" and the semi-colon at the end. For example, "allow 172.99.0.0/24;"

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 Elastic Runtime 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 Elastic Runtime.

To modify the value of the **CF CLI Connection Timeout**, perform the following steps:

1. Select **Advanced Features**.



A screenshot of a web interface showing a single input field with a light gray border. The field contains the number "15". Above the field, the text "CF CLI Connection Timeout" is displayed in a small, dark font.

2. Add a value, in seconds, to the **CF CLI Connection Timeout** field.
3. Click **Save**.

Container-to-Container Networking

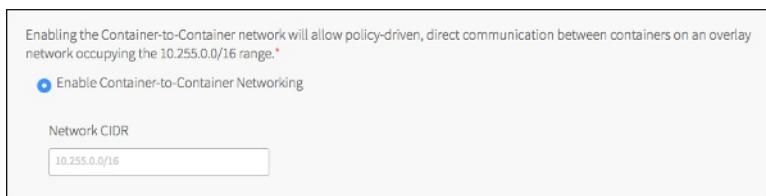
Enabling Container-to-Container Networking allows direct, policy-driven communication between containers on an overlay network occupying a specific range. For more information about Container-to-Container Networking, see the [Administering Cloud Foundry Networking](#) topic.

By default, PCF sets the overlay range to `10.255.0.0/16`, but you can modify this range when you enable Container to Container Networking.

Enable Container-to-Container Networking

To enable Container-to-Container Networking, perform the following steps:

1. Select **Advanced Features**.
2. Select **Enable Container-to-Container Networking**. Optionally, add a range of IP addresses in CIDR format to use instead of the default



A screenshot of a configuration dialog box. At the top, there is a note: "Enabling the Container-to-Container network will allow policy-driven, direct communication between containers on an overlay network occupying the 10.255.0.0/16 range." Below this, there is a radio button labeled "Enable Container-to-Container Networking" which is selected. Underneath, there is a section titled "Network CIDR" with an input field containing the value "10.255.0.0/16".

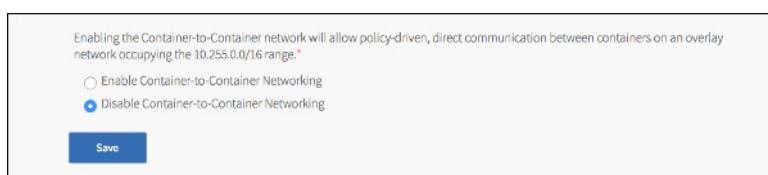
overlay network.

3. Click **Save**.

Disable Container-to-Container Networking

To disable Container-to-Container Networking, perform the following steps:

1. Select **Advanced Features**.
2. Select **Disable Container-to-Container Networking**.



A screenshot of a configuration dialog box. At the top, there is a note: "Enabling the Container-to-Container network will allow policy-driven, direct communication between containers on an overlay network occupying the 10.255.0.0/16 range." Below this, there are two radio buttons: "Enable Container-to-Container Networking" (unchecked) and "Disable Container-to-Container Networking" (checked). At the bottom of the dialog is a blue "Save" button.

3. Click **Save**.

CF API Rate Limiting

Enabling 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.

Enable CF API Rate Limiting

To enable CF API Rate Limiting, perform the following steps:

1. Select **Advanced Features**.
2. Under **Enable CF API Rate Limiting**, select **Enable**.

Enabling CF API Rate Limiting will prevent 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.*

Enable

General Limit *

2000

Unauthenticated Limit *

100

3. 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**.
4. For **Unauthenticated Limit**, enter the number of requests an unauthenticated client is allowed to make over an hour interval. The default value is **100**.
5. Click **Save**.

Disable CF API Rate Limiting

To disable CF API Rate Limiting, perform the following steps:

1. Select **Advanced Features**.
2. Under **Enable CF API Rate Limiting**, select **Disable**.

Enabling CF API Rate Limiting will prevent 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.*

Enable

Disable

3. Click **Save**.

Step 18: Configure Errands

Errands are scripts that Ops Manager runs automatically when it installs or uninstalls a product, such as a new version of Elastic Runtime. There are two types of errands: *post-deploy errands* run after the product is installed, and *pre-delete errands* run before the product is 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. The Elastic Runtime 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.

Errands

Errands are scripts that run at designated points during an installation.

Post-Deploy Errands

Smoke Test Errand

Runs Smoke Tests against your Elastic Runtime installation

On

Apps Manager Errand

Pushes the Pivotal Apps Manager application to your Elastic Runtime installation

On

Notifications Errand

Pushes the Pivotal Notifications application to your Elastic Runtime installation

On

Notifications UI Errand

Pushes the Notifications UI component to your Elastic Runtime installation

On

Pivotal Account Errand

Pushes the Pivotal Account application to your Elastic Runtime installation

On

Autoscaling Errand

Pushes the Pivotal App Autoscaling application to your Elastic Runtime installation

On

Autoscaling Registration Errand

Registers the Autoscaling Service Broker

On

NFS Broker Errand

Pushes the NFS Broker application to your Elastic Runtime installation

On

There are no pre-delete errands for this product.

Save

- **Smoke Test Errand** verifies that your deployment can do the following:
 - Push, scale, and delete apps
 - Create and delete orgs and spaces
- **Apps Manager Errand** deploys the Apps Manager, a dashboard for managing apps, services, orgs, users, and spaces. Until you deploy Apps Manager, you must perform these functions through the CLI. After Apps Manager has been deployed, Pivotal recommends deselecting the checkbox for this errand on subsequent Elastic Runtime deployments. For more information about the 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 Elastic Runtime. See the [Enabling NFS Volume Services](#) topic for more information.

Step 19: 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 IPs as floating IPs.

1. Click **Resource Config**.

Resource Config

JOB	INSTANCES	PERSISTENT DISK TYPE	VM TYPE	FLOATING IPs
Consul	Automatic: 1	Automatic: 1 GB	Automatic: m1.small (cpu: 1, ram: 2 GB, disk: 1)	
NATS	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk: 1)	
etcd	Automatic: 1	Automatic: 1 GB	Automatic: m1.small (cpu: 1, ram: 2 GB, disk: 1)	
Diego BBS	Automatic: 1	Automatic: 1 GB	Automatic: m1.small (cpu: 1, ram: 2 GB, disk: 1)	
File Storage	Automatic: 1	Automatic: 100 GB	Automatic: m1.medium (cpu: 2, ram: 4 GB, disk: 1)	
MySQL Proxy	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk: 1)	
MySQL Server	Automatic: 1	Automatic: 100 GB	Automatic: m1.large (cpu: 4, ram: 8 GB, disk: 1)	
Backup Prepare Node	0	Automatic: 200 GB	Automatic: m1.small (cpu: 1, ram: 2 GB, disk: 1)	
Cloud Controller Database (Postgres)	Automatic: 0	Automatic: 2 GB	Automatic: m1.small (cpu: 1, ram: 2 GB, disk: 1)	
UAA Database (Postgres)	Automatic: 0	Automatic: 10 GB	Automatic: m1.small (cpu: 1, ram: 2 GB, disk: 1)	
UAA	Automatic: 1	None	Automatic: m1.medium (cpu: 2, ram: 4 GB, disk: 1)	
Cloud Controller	Automatic: 1	Automatic: 1 GB	Automatic: m1.medium (cpu: 2, ram: 4 GB, disk: 1)	
HAProxy	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk: 1)	10.85.38.48
Clock Global	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk: 1)	
Cloud Controller Worker	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk: 1)	
Collector	Automatic: 0	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk: 1)	
Diego Brain	Automatic: 1	Automatic: 1 GB	Automatic: m1.small (cpu: 1, ram: 2 GB, disk: 1)	
Diego Cell	Automatic: 3	None	Automatic: m1.xlarge (cpu: 8, ram: 16 GB, disk: 1)	
Doppler Server	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk: 1)	
Loggregator Trafficcontroller	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk: 1)	
Router	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk: 1)	
TCP Router	Automatic: 1	Automatic: 1 GB	Automatic: m1.small (cpu: 1, ram: 2 GB, disk: 1)	10.85.38.50
Push Apps Manager	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk: 1)	
Run Smoke Tests	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk: 1)	
Push Notifications	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk: 1)	
Run Notifications Tests	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk: 1)	
Push Notifications UI	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk: 1)	
Run Notifications-UI tests	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk: 1)	
Push Autoscaling	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk: 1)	
Register Autoscaling Service Broker	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk: 1)	
Destroy autoscaling service broker	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk: 1)	
Run Autoscaling Tests	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk: 1)	
Run CF Acceptance Tests	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk: 1)	
Bootstrap	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk: 1)	
Push Pivotal Account	Automatic: 1	None	Automatic: m1.small (cpu: 1, ram: 2 GB, disk: 1)	

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 20: (Optional) Disable Unused Resources

By default, Elastic Runtime uses an internal filestore and internal databases. If you configure Elastic Runtime to use external resources, you can disable the corresponding system-provided resources in Ops Manager to reduce costs and administrative overhead.

For more information regarding scaling instances, see the [Zero Downtime Deployment and Scaling in CF](#) and the [Scaling Instances in Elastic Runtime](#) topics.

Complete the following procedures to disable specific VMs in Ops Manager:

1. Click **Resource Config**.
2. If you configure Elastic Runtime to use an external S3-compatible filestore, edit the following fields:
 - **File Storage**: Enter in **Instances**.
3. If you configure Elastic Runtime to use an external Relational Database Service (RDS), edit the following fields:
 - **MySQL Proxy**: Enter in **Instances**.
 - **MySQL Server**: Enter in **Instances**.
 - **Cloud Controller Database**: Enter in **Instances**.
 - **UAA Database**: Enter in **Instances**.
4. If you are using an External Load Balancer instead of HAProxy, enter in the **Instances** field for **HAProxy**.
5. Click **Save**.

Step 21: Complete Elastic Runtime 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 screenshot shows the PCF Ops Manager interface. At the top, there's a navigation bar with a teal logo and the text "PCF Ops Manager". On the right, it says "pivotal-cf". Below the navigation bar, a red alert box contains the text "⚠ Please review the errors below" and a bulleted list: "• Cannot reach gateway with IP 10.0.16.1 (ignorable if ICMP is disabled)" and "• Cannot reach DNS with IP 10.0.0.2 (ignorable if ICMP is disabled)". To the right of the alert box are two buttons: "Ignore errors and start the install" (white background, blue text) and "Stop and fix errors" (red background, white text).

3. Elastic Runtime installs. The image shows the **Changes Applied** message that Ops Manager displays when the installation process successfully completes.

The screenshot shows the Pivotal CF Ops Manager interface. At the top, there's a navigation bar with a teal logo and the text "Pivotal CF Ops Manager". Below the navigation bar, a modal window is open with the title "Changes Applied". Inside the modal, there's a green checkmark icon and the text "Ops Manager Director was successfully installed. We recommend that you export a backup of this installation from the actions menu." At the bottom of the modal are two buttons: "Close" (grey background, white text) and "Return to Installation Dashboard" (blue background, white text). The background of the main screen shows a progress bar at 100% and a list of installation steps: "Setting Micro BOSH", "Installing", "Checking Micro BOSH status", and "Logging into director".

Return to [Installing Pivotal Cloud Foundry on OpenStack](#).

Installing Pivotal Cloud Foundry on vSphere

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 the [Pivotal Cloud Foundry Troubleshooting Guide](#).

 **Note:** If you are performing an upgrade to PCF 1.8, see [Upgrading Pivotal Cloud Foundry](#) for critical upgrade information.

Prerequisites

General Requirements

The following are general requirements for deploying and managing a PCF deployment with Ops Manager and Elastic Runtime:

- (**Recommended**) Ability to create a wildcard DNS record to point to your router or load balancer. Alternatively, you can use a service such as [xip.io](#). For example, `203.0.113.0.xip.io`.
- Elastic Runtime 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 router, every application deployed to the `example.com` domain resolves to the IP address of your router.
- (**Recommended**) A network without DHCP available for deploying the Elastic Runtime VMs

 **Note:** If you have DHCP, refer to the Troubleshooting Guide to avoid issues with your installation.

- Sufficient IP allocation:
 - One IP address for each VM instance
 - An additional IP address for each instance that requires static IPs
 - An additional IP address for each errand
 - An additional IP address for each compilation worker: `IPs needed = VM instances + static IPs + errands + compilation workers`

 **Note:** BOSH requires that you allocate a sufficient number of additional dynamic IP addresses when configuring a reserved IP range during installation. BOSH uses these IPs during installation to compile and deploy VMs, install Elastic Runtime, and connect to services. We recommend that you allocate at least 36 dynamic IP addresses when deploying Ops Manager and Elastic Runtime.

- The most recent version of the [Cloud Foundry Command Line Interface \(cf CLI\)](#)
- One or more NTP servers if not already provided by your IaaS

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 Elastic Runtime on vSphere:

- vSphere 6.0, 5.5, or 5.1
- Disk space: 2TB recommended

- Memory: 120GB
- Two public IP addresses: One for Elastic Runtime and one for Ops Manager
- vCPU cores: 80
- Overall CPU: 28 GHz
- vSphere editions: standard and above
- Ops Manager Director 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).
 - Turn hardware virtualization off if your vSphere hosts do not support VT-x/EPT. If you are unsure whether the VM hosts support VT-x/EPT, then you can turn this setting off. If you leave this setting on 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.
- Ops Manager requires read/write permissions to the datacenter level of the [vSphere Inventory Hierarchy](#) to successfully install. Pivotal recommends using the default [VMware Administrator System Role](#) to achieve the appropriate permission level, or a custom role that has all privileges for all objects in the datacenter, including propagating privileges to children. For a complete list of required vSphere privileges, see the [BOSH documentation](#).

 **Note:** For information about how IaaS user roles are configured, refer to the [Pivotal Cloud Foundry IaaS User Role Guidelines](#) topic.

 **Note:** If you are deploying PCF behind a firewall, see the [Preparing Your Firewall for Deploying Pivotal Cloud Foundry](#) topic.

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](#)

Step 2: Install Elastic Runtime

To install Elastic Runtime on vSphere, perform the procedures in the [Configuring Elastic Runtime for vSphere](#) topic.

(Optional) Step 3: 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 Elastic Runtime tile.

Step 4: Create New User Accounts

Once you have successfully deployed PCF, add users to your account. Refer to the [Creating New Elastic Runtime User Accounts](#) topic for more information.

Step 5: Target Your Deployment

Use the cf Command Line Interface (CLI) to target your deployment. Make sure that you have [installed the cf CLI tool](#). Refer to the PCF

documentation for more information about [using the cf command line tool](#)

 **Note:** In Ops Manager, refer to [Elastic Runtime > Credentials](#) for the UAA admin name and password. 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](#)

Deploying Operations Manager to vSphere

Page last updated:

This topic provides instructions for deploying Ops Manager to VMware vSphere.

1. Refer to the [Known Issues](#) topic before getting started.
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.

Ops Manager

Releases: 1.7-alpha5

Release Download Files

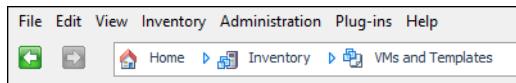
Pivotal Cloud Foundry Ops Manager for vSphere 1.7-alpha5.ova
2.42 GB 1.7-alpha5

Compatibility Matrix Documentation

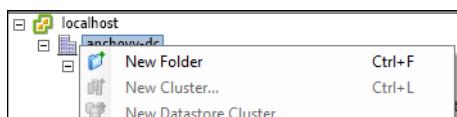
Release Details

RELEASE DATE: 2016-03-04
RELEASE TYPE: Alpha Release

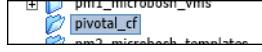
3. Log into vCenter.
4. Select the **VM and Templates** view.



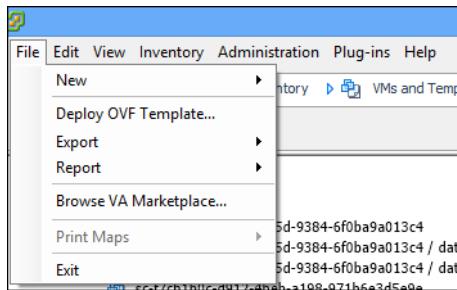
5. Right click on your datacenter and select **New Folder**.



6. Name the folder `pivotal_cf` and select it.



7. Select **File > Deploy OVF Template**.



8. Select the .ova file and click **Next**.

Deploy from a file or URL

Browse...

Enter a URL to download and install the OVF package from the Internet, or specify a location accessible from your computer, such as a local hard drive, a network share, or a CD/DVD drive.

9. Review the product details and click **Next**.

10. Accept the license agreement and click **Next**.

11. Name the virtual machine and click **Next**.

Name:
Pivotal CF vSphere Edition

The name can contain up to 80 characters and it must be unique within the inventory folder.

Inventory Location:

- anchovy-dc
 - birchwood_templates
 - birchwood_vms

Note: The selected folder is the one you created.

12. Select a vSphere cluster and click **Next**.

- anchovy-dc
 - anchovy-cluster

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.

Choose a specific host within the cluster.

On clusters that are configured with vSphere HA or Manual mode vSphere DRS, each virtual machine must be assigned to a specific host, even when powered off.

Select a host from the list below:

Host Name

- 172.16.64.2

15. Select a storage destination and click **Next**.

Select a destination storage for the virtual machine files:

VM Storage Profile:

Name	Drive Type	Capacity	Provisioned	Free	Type	Thin Pro
anchovy-ds	Non-SSD	5.41 TB	1.62 TB	3.98 TB	VMFS5	Supporte
anchovy-ds	SSD	147.00 GB	147.00 GB	214.50 GB	VMFS5	Suppor

16. Select a disk format and click **Next**. For information about disk formats, see [Provisioning a Virtual Disk](#).

Datastore:

Available space (GB):

Thick Provision Lazy Zeroed
 Thick Provision Eager Zeroed
 Thin Provision

17. Select a network from the drop down list and click **Next**.

Source Networks	Destination Networks
Network 1	MattNetwork MattNetwork VM Network VM Network Private

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.

IP Address The IP address for the Pivotal CF Installer. Leave blank if DHCP is desired.
Netmask The netmask for the Pivotal CF Installer's network. Leave blank if DHCP is desired.
Default Gateway The default gateway address for the Pivotal CF Installer's network. Leave blank if DHCP is desired.
DNS The domain name servers for the Pivotal CF Installer (comma separated). Leave blank if DHCP is desired.
NTP Servers Comma-delimited list of NTP servers
Admin Password This password is used to SSH into the Pivotal CF Installer. The username is 'tempest'. Enter password <input type="password"/> Confirm password <input type="password"/>

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 in order to access Ops Manager during the [initial configuration](#).

[Return to the Installing Pivotal Cloud Foundry Guide](#)

Configuring Ops Manager Director for VMware vSphere

Page last updated:

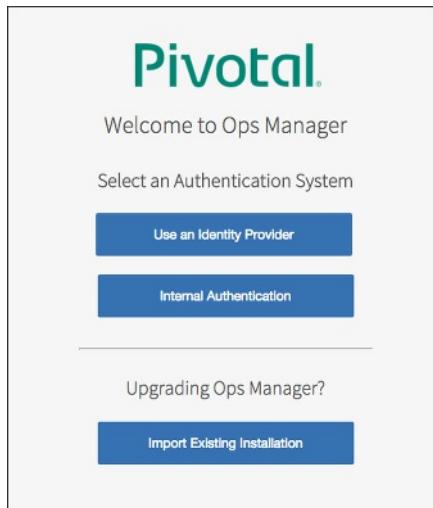
This topic describes how to configure the Ops Manager Director for VMware vSphere.

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 Elastic Runtime for vSphere, vCloud, and vCloud Air](#) 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 are using 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.

The screenshot shows the 'Internal Authentication' setup page. It includes fields for 'Username', 'Password', 'Password confirmation', 'Decryption passphrase', 'Decryption passphrase confirmation', and proxy settings ('Http proxy', 'Https proxy', 'No proxy'). A checkbox for agreeing to the terms and conditions is present, along with a link to the 'End User License Agreement'. A 'Setup Authentication' button is at the bottom.

Step 2: vCenter Config Page

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

The screenshot shows a login page with a 'Welcome!' message. It has fields for 'Email' and 'Password', and a 'SIGN IN' button. There is also a small line of text below the fields.

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



3. Select **vCenter Config**.

[Installation Dashboard](#)

Ops Manager Director

Settings Status Credentials

vCenter Config

vCenter Config

Director Config

Create Availability Zones

Create Networks

Assign AZs and Networks

Security

Resource Config

vCenter Host*
10.87.41.3

vCenter Username*
root

vCenter Password*

[Change](#)

Datacenter Name*
pizza-boxes-dc

Virtual Disk Type*
thin

Ephemeral Datastore Names (comma delimited)*
freenas-ds

The names of the datastores that will store ephemeral VM disks deployed by Ops Manager

NOTE: Removing an Ephemeral Datastore after an initial deploy can result in a system outage and/or data loss.

Persistent Datastore Names (comma delimited)*
freenas-ds

NOTE: Removing a Persistent Datastore after an initial deploy can result in a system outage and/or data loss.

VM Folder*
deadmimes

Template Folder*
deadmimes

Disk path Folder*
deadmimes

[Save](#)

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 used 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.

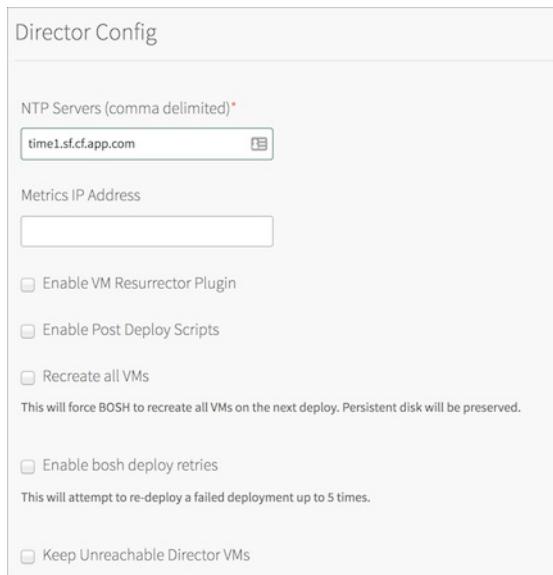
- **Persistent Datastore Names (comma delimited):** The names of the datastores that store persistent VM disks deployed by Ops Manager.
- **VM Folder:** The vSphere datacenter folder (default: `pcf_vms`) where Ops Manager places VMs.
- **Template Folder:** The vSphere datacenter folder (default: `pcf_templates`) 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. 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**.



The screenshot shows the Director Config page with the following fields and settings:

- NTP Servers (comma delimited)*:** time1.sfc.app.com
- Metrics IP Address:** (empty input field)
- Enable VM Resurrector Plugin:**
- Enable Post Deploy Scripts:**
- Recreate all VMs:**
This will force BOSH to recreate all VMs on the next deploy. Persistent disk will be preserved.
- Enable bosh deploy retries:**
This will attempt to re-deploy a failed deployment up to 5 times.
- Keep Unreachable Director VMs:**

2. In the **NTP Servers (comma delimited)** field, enter your NTP server addresses.
3. If you have installed and configured the JMX Bridge product, enter your **Metrics IP Address**.
4. Select the **Enable VM Resurrector Plugin** to enable Ops Manager Resurrector functionality and increase Elastic Runtime availability. For more information, see the [Using Ops Manager Resurrector on VMware vSphere](#) topic.
5. 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.
6. Select **Recreate all VMs** to force BOSH to recreate all VMs on the next deploy. This process does not destroy any persistent disk data.
7. Select **Enable bosh deploy retries** if you want Ops Manager to retry failed BOSH operations up to five times.
8. Select **Keep Unreachable Director VMs** if you want to preserve Ops Manager Director VMs after a failed deployment for troubleshooting purposes.
9. Select **HM Pager Duty Plugin** to enable Health Monitor integration with PagerDuty.

HM Pager Duty Plugin

Service Key*

YOUR-PAGERDUTY-SERVICE-KEY

HTTP Proxy

YOUR-HTTP-PROXY

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

HM Email Plugin

Host*

smtp.example.com

Port*

25

Domain*

cloudfoundry.example.com

From*

user2@example.com

Recipients*

user@example.com, user1@example.com

Username

user

Password

Enable TLS

10. 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.

11. 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

Blobstore Location
 Internal
 S3 Compatible Blobstore

S3 Endpoint*

Bucket Name*

Access Key*

Secret Key*

[Change](#)

V2 Signature
 V4 Signature

Region*

provider.

12. 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**.

Database Location
 Internal
 External MySQL Database

Host*

Port*

Username*

Password*

[Change](#)

Database*

Max Threads

Director Hostname

Save

13. **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.

14. (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](#).

15. 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 Elastic Runtime.

1. Select **Create Availability Zones**.

Create Availability Zones

Availability Zones
Clusters and resource pools to which you will deploy Pivotal products Add

▼ AZ1

Name* A unique name for this availability zone

Cluster*

Resource Pool

Save

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**.

Create Networks

Warning: Pivotal recommends keeping the IP settings throughout the life of your installation. Ops Manager may prevent you from changing them in the future. Contact Pivotal support for help completing such a change.

Verification Settings

Enable ICMP checks

Networks

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

Deadmines [Add Network](#) 

Name* 

Service Network

Subnets

[Add Subnet](#)

vSphere Network Name*

CIDR*

Reserved IP Ranges Ops Manager will not deploy VMs to any IP in this range, e.g. '10.9.9.0-10.9.9.100, 10.9.9.200-10.9.9.255'

DNS* One or more Domain Name Servers used by VMs

Gateway*

Availability Zones* AZ1

[Save](#)

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.



Note: To use the [Single Sign-On for PCF](#) service, you must configure a network with only one subnet.

- 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**.

Assign AZs and Networks

The Ops Manager Director is a single instance.

Choose the availability zone in which to place that instance. It is highly recommended that you backup this VM on a regular basis to preserve settings.

Singleton Availability Zone

AZ1

Network

Deadmines

Save

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**.

The screenshot shows the 'Security' configuration page. It includes a section for 'Trusted Certificates' containing a placeholder for a certificate file (-----BEGIN CERTIFICATE-----). Below this, a note states: 'These certificates enable BOSH-deployed components to trust a custom root certificate.' Under 'Generate VM passwords or use single password for all VMs', there are two radio button options: 'Generate passwords' (selected) and 'Use default BOSH password'. A blue 'Save' button is located at the bottom left.

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 Trusted Registries for running app instances in Docker containers, use this field to enter your certificate for your private Docker Trusted Registry. See the [Using Docker Trusted 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: Resource Config Page

1. Select **Resource Config**.

The screenshot shows the 'Resource Config' configuration page. It lists two jobs: 'Ops Manager Director' and 'Master Compilation Job'. For each job, there are dropdown menus for 'INSTANCES', 'PERSISTENT DISK TYPE', and 'VM TYPE'. The 'INSTANCES' dropdown for 'Ops Manager Director' shows 'Automatic: 1', and for 'Master Compilation Job' shows 'Automatic: 4'. The 'PERSISTENT DISK TYPE' dropdown for 'Ops Manager Director' shows 'Automatic: 50 GB', and for 'Master Compilation Job' shows 'None'. The 'VM TYPE' dropdown for 'Ops Manager Director' shows 'Automatic: medium.disk (cpu: 2, ram: 4 GB)', and for 'Master Compilation Job' shows 'Automatic: large.cpu (cpu: 4, ram: 4 GB, di'. A blue 'Save' button is located at the bottom left.

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.

-
3. Click **Save**.

Step 9: 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 Elastic Runtime for vSphere](#) topic.

Configuring Elastic Runtime for vSphere

Page last updated:

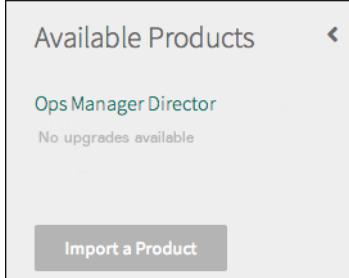
Note: Pivotal Cloud Foundry (PCF) for vCloud Air and vCloud Director is deprecated and availability is restricted to existing customers. Contact [Support](#) for more information.

This topic describes how to configure the Pivotal Elastic Runtime components that you need to run [Pivotal Cloud Foundry](#) (PCF) for VMware vSphere.

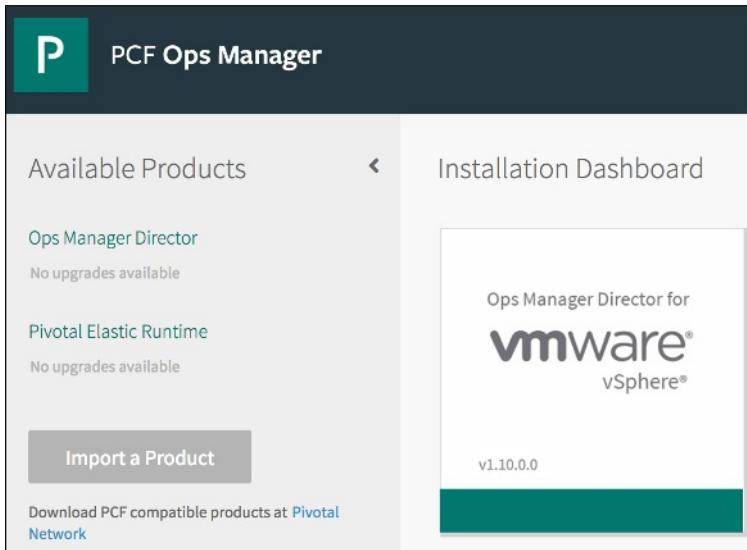
Note: If you plan to [install the IPsec add-on](#), you must do so before installing any other tiles. Pivotal recommends installing IPsec immediately after Ops Manager, and before installing the Elastic Runtime tile.

Step 1: Add Elastic Runtime to Ops Manager

1. Navigate to the [Pivotal Network](#) and click the **Pivotal Cloud Foundry** banner to access the PCF product page. Use the drop-down menu to select an Elastic Runtime release.
2. From the Available Products view, click **Import a Product**.



3. Select the Elastic Runtime `.pivotal` file that you downloaded from Pivotal Network and click **Open**. After the import completes, Elastic Runtime appears in the Available Products view.
4. In the Available Products view, hover over Elastic Runtime and click **Add**.



5. Click the Elastic Runtime 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 Elastic Runtime.

The screenshot shows the "AZ and Network Assignments" section of the Pivotal Elastic Runtime settings. On the left, a sidebar lists several configuration items with checkboxes: "Assign AZs and Networks" (checked), "Domains", "Networking", "Application Containers", "Application Developer Controls", "Application Security Groups", "Authentication and Enterprise SSO", and "Databases". The main panel displays "Place singleton jobs in" with a radio button selected for "first-az". It also shows "Balance other jobs in" with a checked checkbox for "first-az". A "Network" dropdown menu is set to "first-network". At the bottom right is a blue "Save" button.

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.

The screenshot shows the PCF Ops Manager interface with a red warning box. The text inside says: "⚠ Please review the errors below" followed by two bullet points: "Cannot reach gateway with IP 10.0.16.1 (ignorable if ICMP is disabled)" and "Cannot reach DNS with IP 10.0.0.2 (ignorable if ICMP is disabled)". Below the box, it says "All errors will be reverified before installation."

Step 3: Configure Domains

1. Select Domains.

Elastic Runtime hosts applications at subdomains under its apps domain and assigns system components to subdomains under its system domain. You need to configure a wildcard DNS for both the apps domain and system domain. The two domains can be the same, although this is not recommended.

System Domain *

This domain is for system-level PCF components, such as Apps Manager, service brokers, etc. You must set up a wildcard DNS record for this domain that points to your entry point load balancer or HAProxy.

Apps Domain *

Save

2. Enter the system and application domains.

- The **System Domain** defines your target when you push apps to Elastic Runtime.
- The **Apps Domain** defines where Elastic Runtime 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 your own load balancer or the HAProxy load balancer. Find your load balancer type in 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 your subnet.

LOAD BALANCER	ROUTER IP FIELD VALUE	HAProxy IP FIELD VALUE
Your own load balancer	Enter the IP address or addresses for PCF that you registered with your load balancer. Refer to the Using Your Own Load Balancer topic for help using your own load balancer with PCF.	Leave this field blank.
HAProxy load balancer	Leave this field blank.	Enter at least one HAProxy IP address. Point your DNS to this address. For more information, see Configuring SSL/TLS Termination at HAProxy .

For help understanding the Elastic Runtime architecture, refer to the [Architecture](#) topic.

- (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`.
- (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.

Configure security and routing services for your platform. It is usually preferable to use your own load balancer instead of an HAProxy instance as your point-of-entry to the platform.

Router IPs

SSH Proxy IPs

HAProxy IPs

TCP Router IPs

5. Under **Configure the point-of-entry to this environment** choose one of the following options:

 **Note:** For details about the different SSL/TLS termination point options, how they correspond to different points-of-entry for Elastic Runtime, and related certificate requirements, see the [Providing a Certificate for your SSL Termination Point](#) topic.

- **Forward SSL to Elastic Runtime Router:** Select this option if your deployment uses an external load balancer that can forward encrypted traffic to the Elastic Runtime Router, or for a development environment that does not require load balancing. Complete the fields for the **Router SSL Termination Certificate and Private Key** and **Router SSL Ciphers**.

Select one of the following point-of-entry options:^{*}

Forward SSL to Elastic Runtime Router. Assumes an external load balancer is configured to forward encrypted traffic.

Router SSL Termination Certificate and Private Key *

```
-----BEGIN CERTIFICATE-----
[REDACTED]
-----END CERTIFICATE-----
```

Change
Required for SSL encryption between the load balancer and router(s).

Router SSL Ciphers

Forward unencrypted traffic to Elastic Runtime Router. Assumes an external load balancer is configured to forward unencrypted traffic.
 Forward SSL to HA Proxy. Like first option - Assumes an external load balancer is configured to forward encrypted traffic.

- **Forward unencrypted traffic to Elastic Runtime Router:** Select this option if your deployment uses an external load balancer that cannot forward encrypted traffic to the Elastic Runtime Router, or for a development environment that does not require load balancing.

Select one of the following point-of-entry options:^{*}

Forward SSL to Elastic Runtime Router. Assumes an external load balancer is configured to forward encrypted traffic.
 Forward unencrypted traffic to Elastic Runtime Router. Assumes an external load balancer is configured to forward unencrypted traffic.
 Forward SSL to HA Proxy. Like first option - Assumes an external load balancer is configured to forward encrypted traffic.

Disable SSL certificate verification for this environment
 Disable insecure cookies on the Router

- **Forward SSL to HAProxy:** Select this option to use HAProxy as your first point of entry. Complete the fields for **SSL Certificate and Private Key**, and **HAProxy SSL Ciphers**. Select **Disable HTTP traffic to HAProxy** if you want the HAProxy to only allow HTTPS

traffic.

6. 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.

7. Select the **Disable insecure cookies on the Router** checkbox to set the secure flag for cookies generated by the router.
 8. 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](#).

- Disable SSL certificate verification for this environment
- Disable insecure cookies on the Router
- Enable Zipkin tracing headers on the router

- 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.
 - The **Loggregator Port** defaults to `443` if left blank. Enter a new value to override the default.
 - (Optional) Use the **Applications Subnet** field if you need to avoid address collision with a third-party service on the same subnet as your apps. Enter a CIDR subnet mask specifying the range of available IP addresses assigned to your app containers. The IP range must be

different from the network used by the system VMs.

12. (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 . Some configurations, such as networks that use GRE tunnels, may require a smaller MTU value.

Choose whether to enable route services. Route services enable you to proxy requests to your app over TLS to arbitrary URLs before hitting your app.*

Enable route services
 Disable route services

Loggregator Port

Applications Subnet (Only change this if you need to avoid address collision with a third-party service on the same subnet.) *

Applications Network Maximum Transmission Unit (MTU) (in bytes) *

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

14. (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.

15. (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.

Router Timeout to Backends (in seconds) (min: 1) *

Load Balancer Unhealthy Threshold *

Load Balancer Healthy Threshold *

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

Router Max Idle Keepalive Connections (min: 0, max: 50000) *

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

- Select the **Enable TCP Routing** radio button.
- 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 . 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.

Enable TCP requests to your apps via specific ports on the TCP router. You will want to configure a load balancer to forward these TCP requests to the TCP routers. If you do not have a load balancer, then you can also send traffic directly to the TCP router.*

- Select this option if you prefer to enable TCP Routing at a later time
- Enable TCP Routing

TCP Routing Ports (one-time configuration, if you want to update this value you can via the CF CLI) *

1024-1123

- 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. To disable TCP routing, select the **Select this option if you prefer to enable TCP Routing at a later time** radio button. For more information, see the [Configuring TCP Routing in Elastic Runtime](#) topic.

18. Click **Save**.

Step 5: Configure Application Containers

1. Select **Application Containers**.

Enable microservice frameworks, private Docker registries, and other services that support your applications at a container level.

Enable Custom Buildpacks

Allow SSH access to app containers

Enable SSH when an app is created

Private Docker Insecure Registry Whitelist

10.10.10.10:8888,example.com:8888



Docker Images Disk-Cleanup Scheduling on Cell VMs*

- Never clean up Cell disk-space
- Routinely clean up Cell disk-space
- Clean up disk-space once threshold is reached

Threshold of Disk-Used (MB) (min: 1) *

Max Inflight Container Starts *

Save

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 Elastic Runtime 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 Trusted Registries](#) topic for more information.
6. Select your preference for **Docker Images Disk-Cleanup on Cell VMs**. If you choose **Clean up disk-space once threshold is reached** enter a **Threshold of Disk-Used** in megabytes.
7. Optionally, 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. Click **Save**.

Step 6: Configure Application Developer Controls

1. Select **Application Developer Controls**.

Configure restrictions and default settings for applications pushed to Elastic Runtime.

Maximum File Upload Size (MB) (min: 1024, max: 2048) *

Default App Memory (MB) (min: 64, max: 2048) *

Default App Memory Quota per Org (min: 10240, max: 102400) *

Maximum Disk Quota per App (MB) (min: 512, max: 10240) *

Default Disk Quota per App (MB) (min: 512, max: 10240) *

Default Service Instances Quota per Org (min: 0, max: 1000) *

Save

2. Enter your intended maximum file upload size.
3. Enter your default RAM memory allocation per app.
4. Enter your default total RAM memory (RAM) quota per Org. You can change this in the CLI.
5. Enter your maximum and default disk quotas per app.
6. Enter your default service instances quota per Org. You can change this in the CLI.
7. Click **Save**.

Step 7: Review Application Security Group

Setting appropriate [Application Security Groups](#) is critical for a secure deployment. Type in the box to acknowledge that once the Elastic Runtime deployment completes, you will review and set the appropriate application security groups.

Setting appropriate Application Security Groups that control application network policy is the responsibility of the Elastic Runtime administration team. Please refer to the Application Security Groups topic in the Pivotal Cloud Foundry documentation for more detail on completing this activity after the Elastic Runtime deployment completes.

Type X to acknowledge that you understand this message *

Save

Step 8: Configure Authentication and Enterprise SSO

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

Configure your user store access, which can be an internal user store (managed by Cloud Foundry's UAA) or an external user store (LDAP or SAML). You can also adjust the lifetimes of authentication tokens.

Configure your UAA user account store with either internal or external authentication mechanisms*

- Internal UAA (provided by Elastic Runtime; configure your password policy below)

Minimum Password Length *

Minimum Uppercase Characters Required for Password *

Minimum Lowercase Characters Required for Password *

Minimum Numerical Digits Required for Password *

Minimum Special Characters Required for Password *

Number of Months Before Password Expires *

Maximum Password Entry Attempts Allowed *

- SAML Identity Provider

- LDAP Server

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 Elastic Runtime* 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 Elastic Runtime* topic.

- (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.
- (Optional) Customize the text prompts used for username and password from the cf CLI and Apps Manager login popup.
- (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 public IP address(es), append a regular expression or regular expressions to match the public IP address(es).

Apps Manager Access Token Lifetime (in seconds) *

Apps Manager Refresh Token Lifetime (in seconds) *

Cloud Foundry CLI Access Token Lifetime (in seconds) *

Cloud Foundry CLI Refresh Token Lifetime (in seconds) *

Set the lifetime of the refresh token for the Cloud Foundry CLI.

Customize Username Label (on login page) *

Customize Password Label (on login page) *

Proxy IPs Regular Expression *

Save

- Click **Save**.

Step 9: Configure System Databases

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

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:

- Select **Databases**.

Place the databases used by Elastic Runtime components like Cloud Controller and UAA.

Choose the location of your system databases*

- Internal Databases - MySQL and Postgres (the Postgres DBs are not highly-available, but this selection is required if you want to keep your system data from a pre-1.6.0 Elastic Runtime that you upgraded)
- Internal Databases - MySQL (preferred for complete high-availability)
- External Databases (preferred if, for example, you use AWS RDS)

Save

2. Select **Internal Databases - MySQL**.

3. Click **Save**.

Then proceed to [Step 10: \(Optional\) Configure Internal MySQL](#) 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. You can configure a different database provider that provides MySQL support, such as Google Cloud SQL.

To create your Elastic Runtime databases, perform the following steps:

- 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
```

- SSH in to your Ops Manager using the [Ops Manager FQDN](#) and the username `ubuntu`:

```
ssh ubuntu@OPS_MANAGER_FQDN
```

- 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=RDHOSTNAME --user=RDSUSERNAME --password=RDSPASSWORD
```

- Run the following MySQL commands to create databases for the seven Elastic Runtime components that require a relational database:

```
CREATE database uaa;
CREATE database cedb;
CREATE database notifications;
CREATE database autoscale;
CREATE database app_usage_service;
CREATE database routing;
CREATE database diego;
CREATE database account;
CREATE database nfsvolume;
```

- Type `exit` to quit the MySQL client, and `exit` again to close your connection to the Ops Manager VM.
- In Elastic Runtime, select **Databases**.
- Select the **External Databases** option.

Place the databases used by Elastic Runtime components like Cloud Controller and UAA.

Choose the location of your system databases*

- Internal Databases - MySQL and Postgres (the Postgres DBs are not highly-available, but this selection is required if you want to keep your system data from a pre-1.6.0 Elastic Runtime that you upgraded)
- Internal Databases - MySQL (preferred for complete high-availability)
- External Databases (preferred if, for example, you use AWS RDS)

Hostname *

TCP Port *

App Usage Service database username *

App Usage Service database password *

8. Complete the following fields in Elastic Runtime:

Elastic Runtime Field	Notes
Hostname	Specify the hostname of the database server.
TCP Port	Specify the port of the database server.
DATABASE-NAME database username	Specify a unique username that can access this specific database on the database server.
DATABASE-NAME database password	Specify a password for the provided username.

9. Click **Save**.

Step 10: (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 [Proxy](#) section of the *MySQL for PCF* topic for more information.

Only configure this section if you selected Internal Databases - MySQL or Internal Databases - MySQL and Postgres in the previous Databases section.

A proxy tier routes MySQL connections from internal components to healthy cluster nodes. Configure DNS and/or your own load balancer to point to multiple proxy instances for increased availability. TCP healthchecks can be configured against port 1936.

The automated backups functionality works with any S3-compatible file store that can receive your backup files.

MySQL Proxy IPs

MySQL Service Hostname

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.
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.

Replication canary time period *
<input type="text" value="30"/>
Replication canary read delay *
<input type="text" value="20"/>
E-mail address (required) *
<input type="text" value="email@example.com"/>

7. Under **Automated Backups Configuration**, choose one of three options for MySQL backups:
 - **Disable automatic backups of MySQL**
 - **Enable automated backups from MySQL to an S3 bucket or other S3-compatible file stores** saves your backups to an existing Amazon Web Services (AWS) or [Ceph](#) S3-compatible blobstore.

Automated Backups Configuration*

Disable automated backups of MySQL

Enable automated backups from MySQL to an S3 bucket or other S3-compatible file store

S3 Endpoint URL *

S3 Bucket Name *

Bucket Path *

AWS Access Key ID *

AWS Secret Access Key *

Cron Schedule *

Backup All Nodes

Enable automated backups from MySQL to a remote host via SCP

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 **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 automated backups from MySQL to a remote host via SCP** saves your backups to a remote host using secure copy

Automated Backups Configuration*

Disable automated backups of MySQL

Enable automated backups from MySQL to an S3 bucket or other S3-compatible file store

Enable automated backups from MySQL to a remote host via SCP

Hostname *

Port *

Username *

Private key *

Destination directory *

Cron Schedule *

Backup All Nodes

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 Elastic Runtime tile to 0.

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

- 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.

Server Activity Logging*

Disable server activity logging
 Enable server activity logging

Event types *

connect,query

Load Balancer Healthy Threshold *

0

Load Balancer Unhealthy Threshold *

0

Save

9. 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.

10. Click **Save**.

Step 11: 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 Elastic Runtime 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 Elastic Runtime file storage, perform the following steps:

1. In the Elastic Runtime tile, select **File Storage**.

This section determines where you would like to place your Elastic Runtime Cloud Controller's file storage.

Configure your Cloud Controller's filesystem*

- Internal WebDAV (provided by Elastic Runtime)
- External S3-Compatible File Store (if you want to use a service like S3 or Ceph)

URL Endpoint *

`https://s3.amazonaws.com`

Access Key *

XYZ1234567

Secret Key *

.....

S3 Signature Version*

V4 Signature

Region*

US West (N. California)

- Server-side Encryption
(available for AWS S3 only)

Buildpacks Bucket Name *

PcfElasticRuntimeS3Buildpack

Droplets Bucket Name *

PcfElasticRuntimeS3DropletBu

Packages Bucket Name *

PcfElasticRuntimeS3PackagesB

Resources Bucket Name *

PcfElasticRuntimeS3Resources

- External Google Cloud Storage
- External Azure Storage

Save

2. Select the **External S3-Compatible Filestore** option and complete the following fields:

- Enter the **URL Endpoint** for your filestore.
- Enter your **Access Key** and **Secret Key**.
- 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.
- Enter values for the remaining fields as follows:

Ops Manager Field	Description
-------------------	-------------

Buildpacks Bucket Name	S3 bucket for storing app buildpacks.
Droplets Bucket Name	S3 bucket for storing 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	S3 bucket for storing 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	S3 bucket for storing app resources. Pivotal recommends that you use a unique bucket name for app resources, but you can also use the same name as above.

- Click **Save**.

 **Note:** For more information regarding AWS S3 Signatures, see the [Authenticating Requests](#) 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 12: (Optional) Configure System Logging

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

- Select **System Logging**.

Configure system logging. Complete the External Syslog fields only if using an external syslogd server.

Enable Cloud Controller security event logging

External Syslog Aggregator Hostname
 

External Syslog Aggregator Port
 The typical syslogd port is 514. Ensure syslogd is listening on external interfaces.

External Syslog Network Protocol

Syslog Drain Buffer Size (# of messages) *

Save

- 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, and request result, in the Common Event Format (CEF).
- 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 Elastic Runtime network, accept TCP connections, and use the RELP protocol. Ensure your syslog server listens on external interfaces.

- Select an **External Syslog Network Protocol** to use when forwarding logs.
- 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**.

Step 13: (Optional) Customize Apps Manager

The **Custom Branding** and **Apps Manager** sections customize the appearance and functionality of Apps Manager. Refer to [Custom Branding](#) and [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.

Customize colors, images, and text for Apps Manager and the Cloud Foundry login portal.

Company Name

Accent Color

Main Logo (PNGs only)

Square Logo/Favicon (PNGs only)

Footer Text
 Defaults to 'Pivotal Software Inc. All rights reserved.'

Footer Links
You may configure up to three links in the Apps Manager footer Add

Classification Header/Footer Background Color

Classification Header/Footer Text Color

Classification Header Content

Classification Footer Content

Save

2. Click **Save** to save your settings in this section.
3. Select **Apps Manager**. Use this section to configure page names and sidebar links in the **Apps Manager** and **Marketplace** pages.

Configure Apps Manager

Display Marketplace Service Plan Prices

Supported currencies as json *****

 Define the currency codes and associated symbols (defaults to "{ \"usd\": \"\$\", \"eur\": \"€\" }" if blank)

Product Name

Marketplace Name

Customize Sidebar Links

You may configure up to 10 links in the Apps Manager sidebar

Add 

▶ Marketplace	
▶ Docs	
▶ Tools	

Save

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

Step 14: (Optional) Configure Email Notifications

Elastic Runtime 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**.

Configure Simple Mail Transfer Protocol for the Notifications application to send email notifications about your deployment. This application is deployed as an errand in Elastic Runtime. If you do not need this service, leave this section blank and disable the Notifications and Notifications UI errands.

From Email

Address of SMTP Server

Port of SMTP Server

SMTP Server Credentials

[Change](#)

SMTP Enable Automatic STARTTLS

SMTP Authentication Mechanism*

SMTP CRAMMD5 secret

Save

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 15: (Optional) Add CCDB Restore Key

Perform this step if all of the following are true:

- You deployed Elastic Runtime previously.
- You then stopped Elastic Runtime or it crashed.
- You are re-deploying Elastic Runtime with a backup of your Cloud Controller database.

1. Click **Restore CCDB Encryption Key**.
2. Enter your **Cloud Controller DB Encryption Key**.

You must enter your original database encryption key here once you have deployed Elastic Runtime for the first time. Anytime thereafter your encryption key will remain the same, if you specify the same key here. This is important for backup and restore of data as well as upgrades.

Cloud Controller DB Encryption Key

...

Leave blank unless restoring a Cloud Controller Database exported from a previous installation.

Save

See [Backing Up Pivotal Cloud Foundry](#) for more information.

Step 16: Configure Smoke Tests

The Smoke Tests errand runs basic functionality tests against your Elastic Runtime 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 org and space**, which creates an ad-hoc org and space for running smoke tests and deletes them afterwards. Otherwise, select **Specified org and space** and complete the fields to specify where you want to run smoke tests.

Specify a Cloud Foundry organization and space where smoke tests can run if in the future you delete your Elastic Runtime deployment domains.

Choose whether to create a new org and space on demand for smoke tests*

- Temporary org and space (This org and space are deleted after smoke tests finish.)
 Specified org and space (The org and space must have a domain available for routing.)

Organization *

Space *

Domain *

Save

3. Click **Save**.

Step 17: (Optional) Enable Advanced Features

The Advanced Features section of Elastic Runtime 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

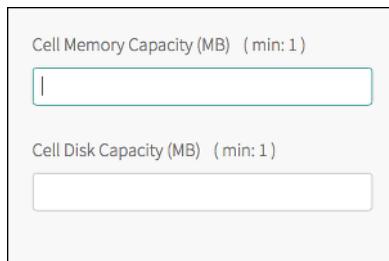
If your apps do not use the full allocation of disk space and memory set in the **Resource Config** tab, you might want use this feature. These fields control the amount to overcommit disk and memory resources to each Diego Cell VM.

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**.



The screenshot shows a form with two input fields. The first field is labeled "Cell Memory Capacity (MB) (min: 1)" with a text input box containing a single character. The second field is labeled "Cell Disk Capacity (MB) (min: 1)" with an empty text input box.

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

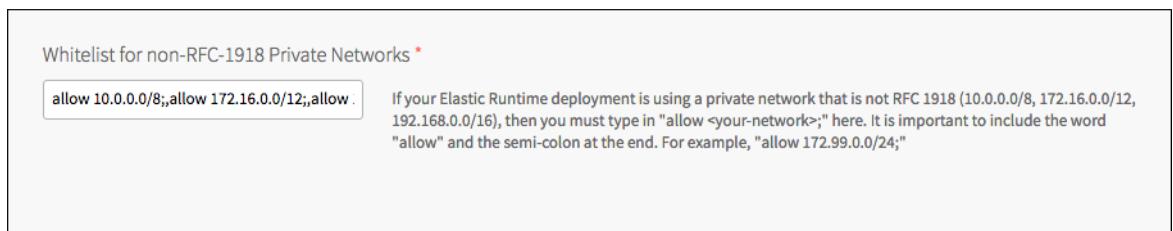
Some private networks require extra configuration so that internal file storage (WebDAV) can communicate with other PCF processes.

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.



The screenshot shows a text input field for the whitelist. It contains the text "allow 10.0.0.0/8;allow 172.16.0.0/12;allow ". To the right of the input field, there is a note: "If your Elastic Runtime deployment is using a private network that is not RFC 1918 (10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16), then you must type in "allow <your-network>;" here. It is important to include the word "allow" and the semi-colon at the end. For example, "allow 172.99.0.0/24;"

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 Elastic Runtime 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 Elastic Runtime.

To modify the value of the **CF CLI Connection Timeout**, perform the following steps:

1. Select **Advanced Features**.

A screenshot of a configuration dialog titled "CF CLI Connection Timeout". It contains a single input field with the value "15" entered.

2. Add a value, in seconds, to the **CF CLI Connection Timeout** field.
3. Click **Save**.

Container-to-Container Networking

Enabling Container-to-Container Networking allows direct, policy-driven communication between containers on an overlay network occupying a specific range. For more information about Container-to-Container Networking, see the [Administering Cloud Foundry Networking](#) topic.

By default, PCF sets the overlay range to `10.255.0.0/16`, but you can modify this range when you enable Container to Container Networking.

Enable Container-to-Container Networking

To enable Container-to-Container Networking, perform the following steps:

1. Select **Advanced Features**.
2. Select **Enable Container-to-Container Networking**. Optionally, add a range of IP addresses in CIDR format to use instead of the default

A screenshot of a configuration dialog for enabling Container-to-Container Networking. It includes a note about enabling the network, a radio button for "Enable Container-to-Container Networking" (which is selected), and a text input field for "Network CIDR" containing the value "10.255.0.0/16".

overlay network.

3. Click **Save**.

Disable Container-to-Container Networking

To disable Container-to-Container Networking, perform the following steps:

1. Select **Advanced Features**.
2. Select **Disable Container-to-Container Networking**.

A screenshot of a configuration dialog for disabling Container-to-Container Networking. It includes a note about enabling the network, two radio buttons ("Enable Container-to-Container Networking" and "Disable Container-to-Container Networking", with the latter selected), and a "Save" button at the bottom.

3. Click **Save**.

CF API Rate Limiting

Enabling 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.

Enable CF API Rate Limiting

To enable CF API Rate Limiting, perform the following steps:

1. Select **Advanced Features**.
2. Under **Enable CF API Rate Limiting**, select **Enable**.

Enabling CF API Rate Limiting will prevent 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.*

Enable

General Limit *

2000

Unauthenticated Limit *

100

3. 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**.
4. For **Unauthenticated Limit**, enter the number of requests an unauthenticated client is allowed to make over an hour interval. The default value is **100**.
5. Click **Save**.

Disable CF API Rate Limiting

To disable CF API Rate Limiting, perform the following steps:

1. Select **Advanced Features**.
2. Under **Enable CF API Rate Limiting**, select **Disable**.

Enabling CF API Rate Limiting will prevent 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.*

Enable

Disable

3. Click **Save**.

Step 18: Configure Errands

Errands are scripts that Ops Manager runs automatically when it installs or uninstalls a product, such as a new version of Elastic Runtime. There are two types of errands: *post-deploy errands* run after the product is installed, and *pre-delete errands* run before the product is 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. The Elastic Runtime 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.

Errands

Errands are scripts that run at designated points during an installation.

Post-Deploy Errands

Smoke Test Errand

Runs Smoke Tests against your Elastic Runtime installation

On

Apps Manager Errand

Pushes the Pivotal Apps Manager application to your Elastic Runtime installation

On

Notifications Errand

Pushes the Pivotal Notifications application to your Elastic Runtime installation

On

Notifications UI Errand

Pushes the Notifications UI component to your Elastic Runtime installation

On

Pivotal Account Errand

Pushes the Pivotal Account application to your Elastic Runtime installation

On

Autoscaling Errand

Pushes the Pivotal App Autoscaling application to your Elastic Runtime installation

On

Autoscaling Registration Errand

Registers the Autoscaling Service Broker

On

NFS Broker Errand

Pushes the NFS Broker application to your Elastic Runtime installation

On

There are no pre-delete errands for this product.

Save

- **Smoke Test Errand** verifies that your deployment can do the following:
 - Push, scale, and delete apps
 - Create and delete orgs and spaces
- **Apps Manager Errand** deploys the Apps Manager, a dashboard for managing apps, services, orgs, users, and spaces. Until you deploy Apps Manager, you must perform these functions through the of CLI. After Apps Manager has been deployed, Pivotal recommends deselecting the checkbox for this errand on subsequent Elastic Runtime deployments. For more information about the 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 Elastic Runtime. See the [Enabling NFS Volume Services](#) topic for more information.

Step 19: (Optional) Configure Resources

Note: Ops Manager 1.9 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**.

Scale the number of instances in order to reduce resources and configure your deployment.

JOB	INSTANCES	PERSISTENT DISK TYPE	VM TYPE
Consul	Automatic: 1	Automatic: 1 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)
NATS	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)
etcd	Automatic: 1	Automatic: 1 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)
Diego BBS	Automatic: 1	Automatic: 1 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)
File Storage	Automatic: 1	Automatic: 100 GB	Automatic: medium.mem (cpu: 1, ram: 8 GB, disk: 100 GB)
MySQL Proxy	Automatic: 1	None	Automatic: small (cpu: 1, ram: 2 GB, disk: 4 GB)
MySQL Server	Automatic: 1	Automatic: 100 GB	Automatic: large.disk (cpu: 2, ram: 8 GB, disk: 100 GB)
Backup Prepare Node	0	Automatic: 200 GB	Automatic: small (cpu: 1, ram: 2 GB, disk: 4 GB)
Cloud Controller Database (Postgres)	Automatic: 0	Automatic: 2 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)
UAA Database (Postgres)	Automatic: 0	Automatic: 10 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)
UAA	Automatic: 1	None	Automatic: medium.disk (cpu: 2, ram: 4 GB, disk: 10 GB)
Cloud Controller	Automatic: 1	Automatic: 1 GB	Automatic: medium.disk (cpu: 2, ram: 4 GB, disk: 1 GB)
HAProxy	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)
Clock Global	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)
Cloud Controller Worker	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)
Collector	Automatic: 0	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)
Diego Brain	Automatic: 1	Automatic: 1 GB	Automatic: small (cpu: 1, ram: 2 GB, disk: 4 GB)
Diego Cell	Automatic: 3	None	Automatic: xlarge.disk (cpu: 4, ram: 16 GB, disk: 100 GB)

Doppler Server	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)
Loggregator	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)
Trafficcontroller	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)
Router	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)
TCP Router	Automatic: 1	Automatic: 1 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)
Push Apps Manager	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)
Run Smoke Tests	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)
Push Notifications	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)
Run Notifications Tests	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)
Push Notifications UI	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)
Run Notifications-UI tests	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)
Push Autoscaling	Automatic: 1	None	Automatic: nano (cpu: 1, ram: 512 MB, disk: 1 GB)
Register Autoscaling Service Broker	Automatic: 1	None	Automatic: nano (cpu: 1, ram: 512 MB, disk: 1 GB)
Destroy autoscaling service broker	Automatic: 1	None	Automatic: nano (cpu: 1, ram: 512 MB, disk: 1 GB)
Run Autoscaling Tests	Automatic: 1	None	Automatic: nano (cpu: 1, ram: 512 MB, disk: 1 GB)
Run CF Acceptance Tests	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)
Bootstrap	Automatic: 1	None	Automatic: small (cpu: 1, ram: 2 GB, disk: 4 GB)
Push Pivotal Account	Automatic: 1	None	Automatic: small (cpu: 1, ram: 2 GB, disk: 4 GB)

Save

By default, Elastic Runtime uses an internal filestore and internal databases. If you configure Elastic Runtime to use external resources, you can disable the corresponding system-provided resources in Ops Manager to reduce costs and administrative overhead.

For more information regarding scaling instances, see the [Zero Downtime Deployment and Scaling in CF](#) and the [Scaling Instances in Elastic Runtime](#) topics.

Complete the following procedures to disable specific VMs in Ops Manager:

1. Click **Resource Config**.
2. If you configure Elastic Runtime to use an external S3-compatible filestore, edit the following fields:
 - File Storage:** Enter in **Instances**.
3. If you configure Elastic Runtime to use an external Relational Database Service (RDS), edit the following fields:
 - MySQL Proxy:** Enter in **Instances**.
 - MySQL Server:** Enter in **Instances**.
 - Cloud Controller Database:** Enter in **Instances**.
 - UAA Database:** Enter in **Instances**.
4. If you are using an External Load Balancer instead of HAProxy, enter in the **Instances** field for **HAProxy**.
5. Click **Save**.

Step 20: Configure Stemcell

1. Select **Stemcell**. This page displays the stemcell version that shipped with Ops Manager.

Stemcell

A stemcell is a template from which Ops Manager creates the VMs needed for a wide variety of components and products.

cf requires BOSH stemcell version 3262 ubuntu-trusty

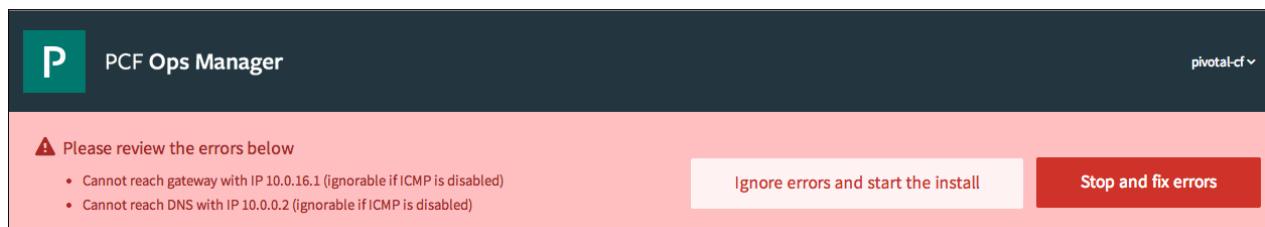
✓ Using [boshs-stemcell-3262.7-vsphere-esxi-ubuntu-trusty-go_agent.tgz](#)

[Import Stemcell](#)

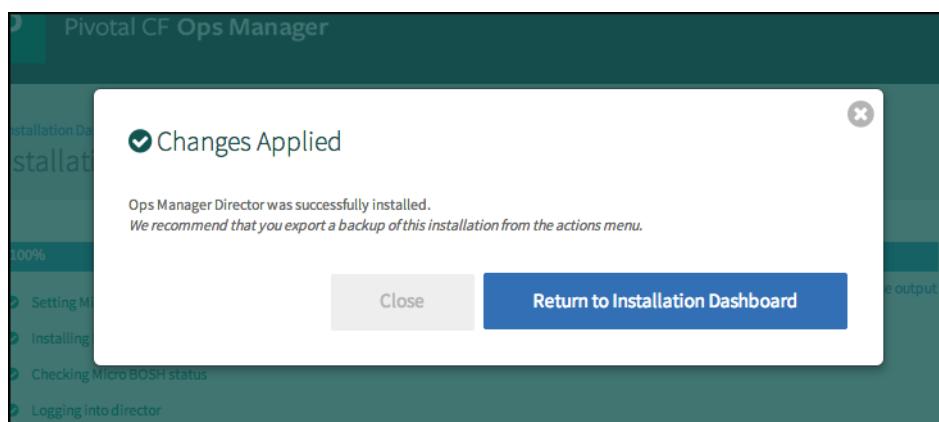
You can also use this page to import a new stemcell version. You only need to import a new Stemcell if your Ops Manager does not already have the Stemcell version required by Elastic Runtime.

Step 21: Complete the Elastic Runtime 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.



Provisioning a Virtual Disk in vSphere

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 as 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:

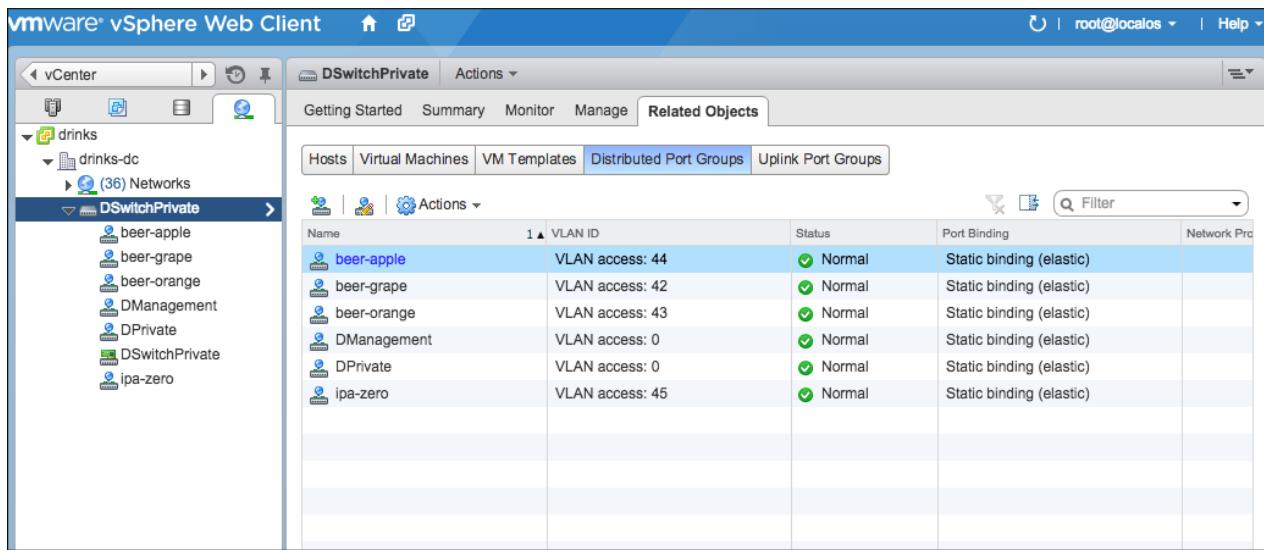
Refer to the procedure in this topic to use Ops Manager with the Cisco Nexus 1000v Switch. First, configure Ops Manager through Step 4 in [Configuring Ops Manager Director for VMware vSphere](#). Then configure your network according to the following steps.

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.

The screenshot shows the 'Create Networks' configuration page for the 'default' network. The left sidebar lists configuration sections: vCenter Config, Director Config, Create Availability Zones, Create Networks (which is selected), Assign AZs and Networks, Security, and Resource Config. The main area is titled 'Create Networks' and contains the following fields:

- Verification Settings:** A checked checkbox for 'Enable ICMP checks'.
- Networks:** A section for defining IP ranges. It includes a dropdown for 'default', a 'Name*' field containing 'default' with a note 'A unique name for this network', and an 'Add Network' button.
- Subnets:** Fields for 'vSphere Network Name*', 'CIDR*', 'Reserved IP Ranges', 'DNS*', and 'Gateway*'. There is also a checkbox for 'Availability Zones*' with the option 'first-az'.
- Buttons:** A 'Save' button at the bottom and 'Add Subnet' buttons for both the Networks and Subnets sections.

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.



- 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.

[Installation Dashboard](#)

Ops Manager Director

- Settings
- Status
- Credentials

vCenter Config

Director Config

Create Availability Zones

Create Networks

Assign AZs and Networks

Security

Resource Config

Create Networks

Warning: Pivotal recommends keeping the IP settings throughout the life of your installation. Ops Manager may prevent you from changing them in the future. Contact Pivotal support for help completing such a change.

Verification Settings

Enable ICMP checks

Networks

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

Add Network

YOUR_NETWORK_NAME

Name*

Subnets

Add Subnet

vSphere Network Name*

The name of the network as it appears in vCenter

CIDR*

Reserved IP Ranges

DNS*

Gateway*

Availability Zones*

Save

6. Click **Save**.

7. Return to [Configuring Ops Manager Director for VMware vSphere](#) to complete the Ops Manager installation.

Using Ops Manager Resurrector on VMware vSphere

Page last updated:

The Ops Manager Resurrector increases [Pivotal Cloud Foundry](#) (PCF) Elastic Runtime 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 Elastic Runtime 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:

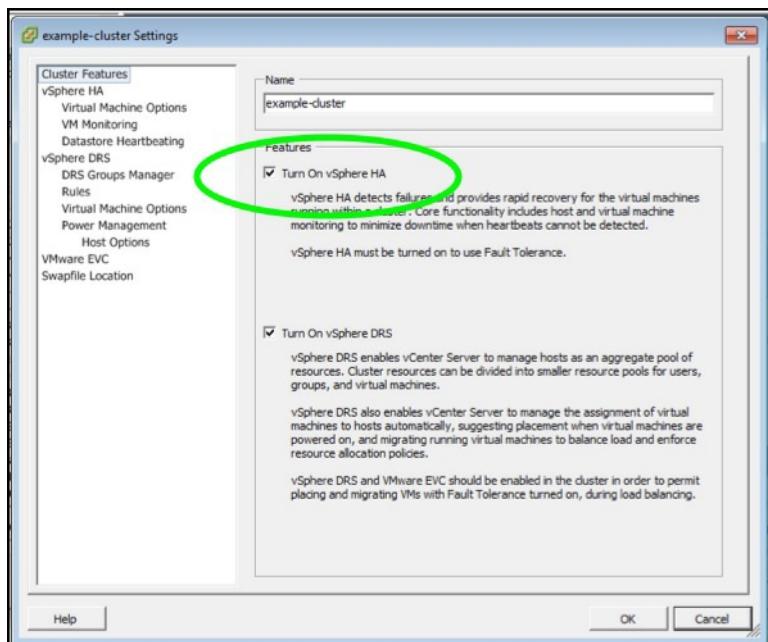
- The Resurrector does not monitor or protect the Ops Manager VM or the BOSH Director VM.
- The Resurrector might not be able to resolve issues caused by the loss of an entire host.
- The Resurrector does not monitor or protect data storage.

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**.

Installation Dashboard

Ops Manager Director

Settings Status Credentials

vCenter Config Director Config Create Availability Zones Create Networks Assign AZs and Networks Security Resource Config

Director Config

NTP Servers (comma delimited)*
time1.sf.cf-app.com

Metrics IP Address

Enable VM Resurrector Plugin

Recreate all VMs

This will force BOSH to recreate all VMs on the next deploy. Persistent disk will be preserved.

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 Elastic Runtime.

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 Elastic Runtime 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 Elastic Runtime 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 Elastic Runtime 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
<code>*.cf</code>	A	<code>203.0.113.1</code>	<code>example.com</code>

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 Elastic Runtime 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 Elastic Runtime 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 Elastic Runtime 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
<code>*.cf</code>	A	<code>198.51.100.1</code>	<code>example.com</code>

3. From the PCF Ops Manager Dashboard, click on the Elastic Runtime 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](#)

Understanding Availability Zones in VMware Installations

Page last updated:

 **Note:** Pivotal Cloud Foundry (PCF) for vCloud Air and vCloud Director is deprecated and availability is restricted to existing customers. Contact [Support](#) for more information.

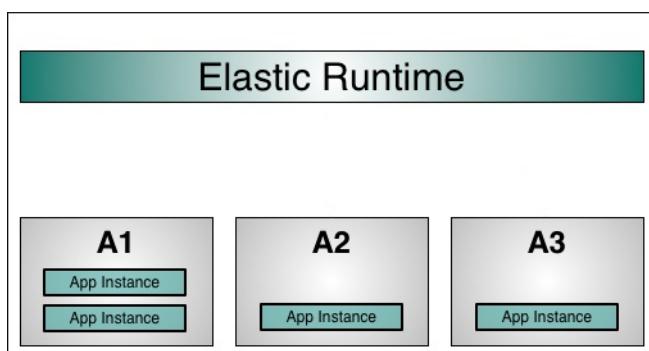
Pivotal defines an Availability Zone (AZ) as an operator-assigned, functionally independent segment of network infrastructure. In cases of partial infrastructure failure, [Pivotal Cloud Foundry](#) (PCF) Elastic Runtime distributes and balances all instances of running applications across remaining AZs. Strategic use of Availability Zones contributes to the fault tolerance and high availability of a Elastic Runtime deployment.

Elastic Runtime on VMware vSphere supports distributing deployments across multiple AZs. See the section on AZs in [Configuring Ops Manager Director for VMware vSphere](#).

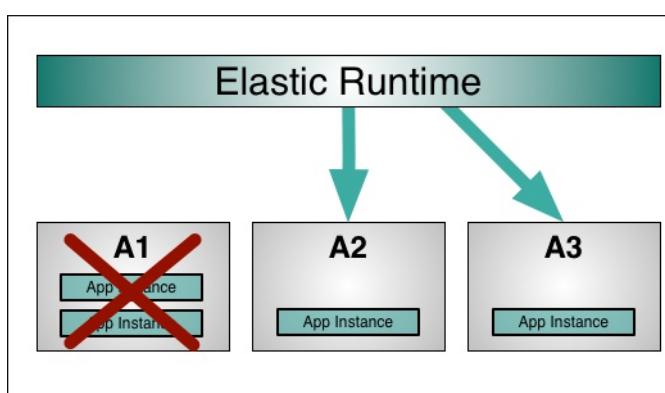
It is recommended that customers use three Availability Zones to operate a highly available installation of Elastic Runtime.

Balancing Across AZs During Failure: Example Scenario

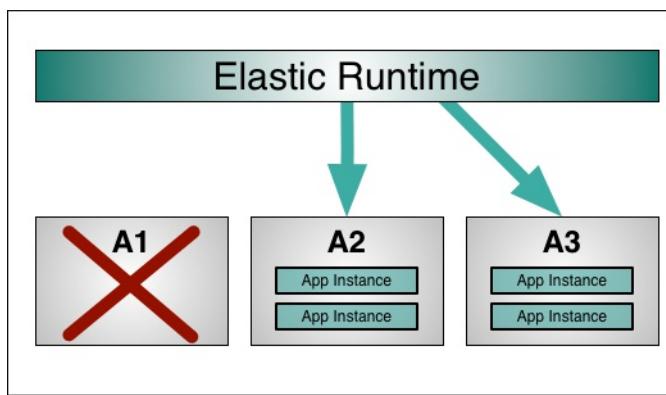
An operator scales an application to four instances in an Elastic Runtime environment distributed across three availability zones: A1, A2, and A3. The environment allocates the instances according to the [Diego Auction](#).



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, Elastic Runtime balances new instances of the application across the remaining availability zones:



Upgrading Pivotal Cloud Foundry

Page last updated:

This topic describes upgrading Pivotal Cloud Foundry (PCF) to v1.10. The upgrade procedure below describes upgrading Pivotal Cloud Foundry Operations Manager (Ops Manager), Pivotal Elastic Runtime, 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](#) 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.

 **Note:** Previous versions of PCF used SHA-1, while PCF v1.10 uses SHA-2 by default. However, you cannot obtain SHA-2 by upgrading to PCF v1.10. Instead, you must perform a fresh install by following the steps in [Installing Pivotal Cloud Foundry](#).

Before You Upgrade

This section contains important preparation steps, **Preps**, that you must follow before beginning an upgrade to Ops Manager v1.10. Failure to follow these instructions may jeopardize your existing deployment data and cause your upgrade to fail.

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 Elastic Runtime file storage is performant enough to handle the upgrade. For more information on 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.

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: Review Partner Service Tiles

Some partner service tiles may be incompatible with PCF v1.10. 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 v1.10, review the appropriate partners services release documentation at <http://docs.pivotal.io>, or contact the partner organization that produces the service tile.

Prep 4: 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 Elastic Runtime v1.10.x.

Prep 5: 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 v1.7 and you are upgrading to v1.10, you must sequentially install v1.8 and v1.9 before proceeding with the upgrade to v1.10.
2. **IMPORTANT:** Back up all critical data prior to upgrading to Ops Manager v1.10. For example, to back up a v1.9 environment, follow the instructions in the v1.9 [Backing Up Pivotal Cloud Foundry](#) topic.
3. 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.
4. 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 Elastic Runtime. 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**.

5. 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**.
6. 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 6: Upgrade and Configure RabbitMQ for PCF

If your PCF deployment contains **RabbitMQ for PCF**, download and upgrade RabbitMQ for PCF to v1.7.13 or later.

For upgrade instructions, see the [RabbitMQ for PCF documentation](#).

Ensure that you complete the following as part of the upgrade:

- Firewall rules on your Rabbit VM instances must allow inbound traffic on port **8301**.
- In the **RabbitMQ metrics configuration** screen, deselect the **Use non-secure communication for metrics** checkbox. This checkbox must be unchecked for metrics to be emitted to the Firehose.

Prep 7: Upgrade and Configure Redis for PCF

If your PCF deployment contains **Redis for PCF**, download and upgrade to Redis for PCF v1.7.3 or later.

For upgrade instructions, see the [Redis for PCF documentation](#).

Ensure that you complete the following as part of the upgrade:

- When performing the upgrade process, ensure that persistent disk is set to 3.5x the amount of RAM.
- Firewall rules on your Redis VM instances must allow inbound traffic on port `8301`.
- In the **Redis metrics configuration** screen, deselect the **Use non-secure communication for metrics** checkbox. This checkbox must be unchecked for metrics to be emitted to the Firehose.

Prep 8: Check OS Compatibility of PCF and BOSH-Managed Add-Ons

Before upgrading to PCF v1.10, 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 [PCF Runtime for Windows](#) 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 on configuring BOSH add-on manifests, see the [BOSH documentation](#).

Prep 9: Check System Health Before Upgrade

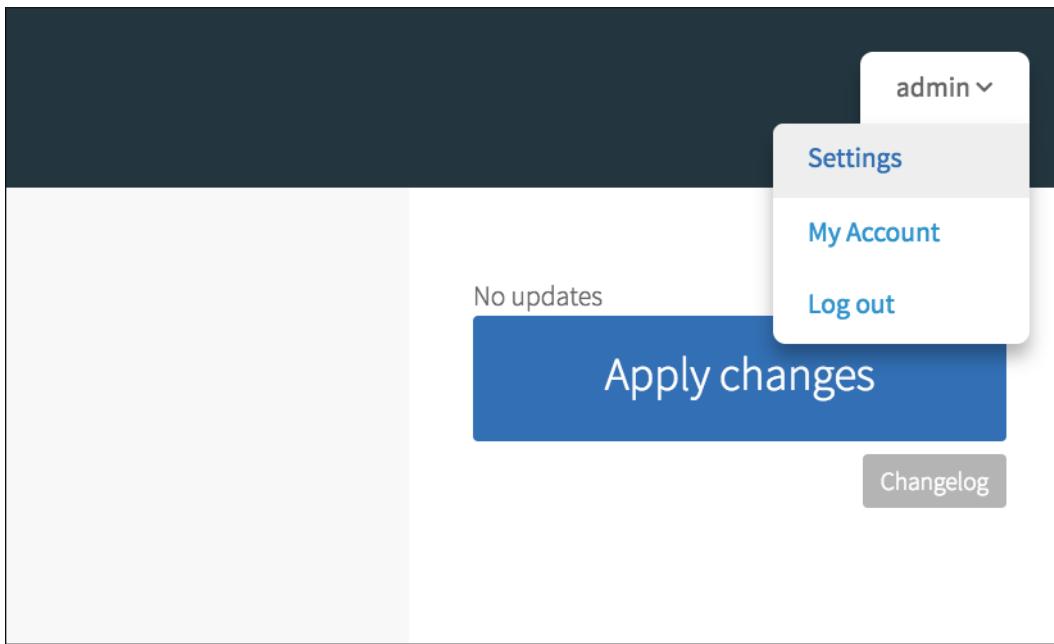
1. Run `bosh cloudbuild` to confirm that the VMs are healthy. For more information, see the [BOSH Cloudbuild](#) topic.
2. Check the system health of installed products. In the **Installation Dashboard**, select the **Status** tab for each service tile. Confirm that all jobs are healthy.
3. (Optional) 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 v1.10

Step 1: Export Your Installation

1. In your Ops Manager v1.9.x **Installation Dashboard**, click the account dropdown and select **Settings**.



- On the **Settings** screen, select **Export Installation Settings** from the left menu, and click the **Export Installation Settings** button.

PCF Ops Manager

[Installation Dashboard](#)

[Settings](#)

Decryption Passphrase

Authentication Method

External API Access

Proxy Settings

Export Installation Settings

Advanced

Export Installation

⚠️ Upgrading Ops Manager may block the upgrade path for certain products.
Please read [Upgrading Ops Manager](#) and verify an upgrade path exists for your installed products.

Export Installation Settings

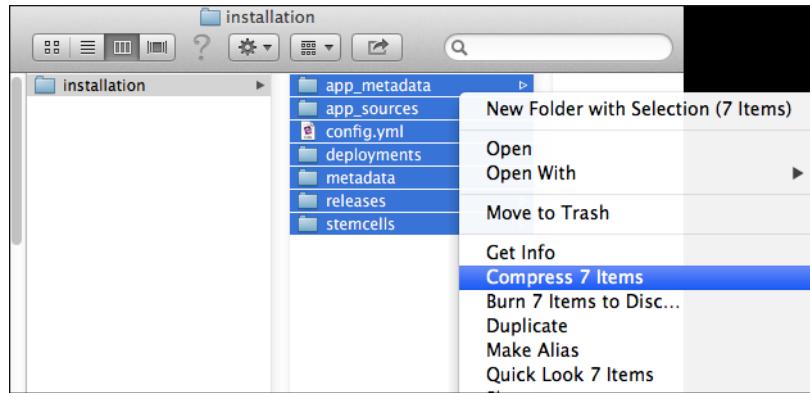
This exports the current PCF installation with all of its assets.

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.

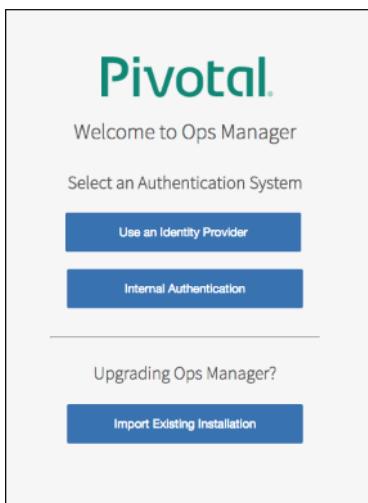
- Some browsers do not provide feedback on the status of the export process and might appear to hang.

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 v1.10

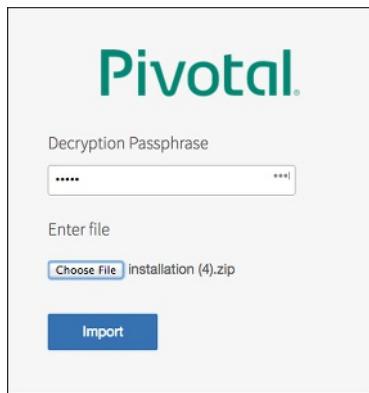
- Download the Ops Manager VM Template v1.10.x from the [Pivotal Network](#) site.
- Record the FQDN address of the existing Ops Manager VM.
- To avoid conflicts, power off the existing Ops Manager VM.
- Deploy the new Ops Manager VM by following the steps in one of these topics:
 - [AWS: Launching an Ops Manager Director Instance on AWS](#)
 - [Azure: Launching an Ops Manager Director Instance with an ARM Template](#)
 - [GCP: Launching an Ops Manager Director Instance on GCP](#)
 - [OpenStack: Provisioning the OpenStack Infrastructure](#)
 - [vSphere: Deploying Operations Manager to vSphere](#)
- When redirected to the **Welcome to Ops Manager** page, select **Import Existing Installation**.



- 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.

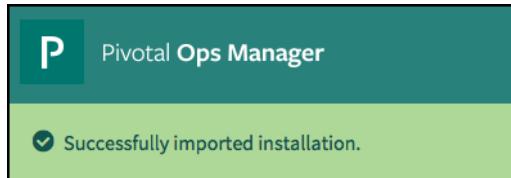
- Click **Choose File** and browse to the installation ZIP file exported in Step 1 above.



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 Elastic Runtime and Product Tiles

1. After upgrading to Ops Manager v1.10, 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 Elastic Runtime and Other Pivotal Cloud Foundry Products](#) topic for more information.

Step 4: 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 (v1.9.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 (v1.9.x) Ops Manager VM.

This completes the upgrade to Ops Manager v1.10.

After You Upgrade

Check Loggregator Throughput and Scale Dopplers if Necessary

Since Loggregator now uses the gRPC protocol, your deployment may see an increase in Loggregator message throughput.

Pivotal recommends that you monitor throughput in case you need to scale up Dopplers.

As an example, the following metrics equation measures the percentage of message loss in a deployment using a DataDog nozzle.

```
100 * ( rate(sum:datadog.nozzle.doppler.loggregator.ingress{*}) +  
rate(sum:datadog.nozzle.DopplerServer.listeners.receivedEnvelopes{*}) ) /  
(rate(sum:datadog.nozzle.MetronAgent.DopplerForwarder.sentMessages{*}) +  
rate(sum:datadog.nozzle.metron.loggregator.egress{*}) )
```

To measure Loggregator message loss in your deployment, substitute the nozzle configured for your deployment in the above equation.

If you notice an increase in loss percentage, then scale up your Doppler instances.

For more information on monitoring and scaling Loggregator, see the following documents:

- [How-to-Calculate-the-Loggagators-Message-Throughput](#)
- [Configuring System Logging in Elastic Runtime](#)

Upgrade cf CLI

To avoid connection issues when tailing logs on the command line, users should upgrade to cf CLI v6.23 or later. This version is the minimum recommended version for use with PCF v1.10.

After you upgrade the cf CLI, if you still encounter a connection issue, make sure you log in and log out again using `cf logout` and `cf login`.

Understanding the Effects of Single Components on a Pivotal Cloud Foundry Upgrade

Page last updated:

The **Resource Config** page of Pivotal Elastic Runtime 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	✓	✓	
3	etcd Server	✓	✓	Several components rely upon etcd for configuration and persistence. If etcd is unavailable, these components may fail in expected ways as they are not able to access their configuration or persisted data.
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			
8	Cloud Controller Database (Postgres)		✓	
9	UAA Database		✓	

	(Postgres)			
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.
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	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 [Elastic Runtime 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. 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.

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 Elastic Runtime 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 Elastic Runtime 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 on 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	10% of total instances	Yes
PCF 1.9.8 +	200	Yes	10% of total instances	Yes
PCF 1.10.0 and later	200	Yes	10% 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. If this 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.

For more information on how Diego cells are upgraded, see the [Managing Diego Cell Limits During an Upgrade](#) topic.

Related Links

- [How to use Elastic Runtime blob storage data](#)
- [Upgrading Pivotal Cloud Foundry](#)
- [Managing Diego Cell Limits During an Upgrade](#)

Upgrading Elastic Runtime and Other Pivotal Cloud Foundry Products

Page last updated:

This topic describes how to upgrade to a point release of Elastic Runtime (ERT) and other product tiles without upgrading Ops Manager. For example, use this topic to upgrade from ERT 1.9.0 to 1.9.1. You might need to perform this upgrade if a security update for ERT is released, or if new features are introduced in a point release of a product tile.

For Elastic Runtime component and version information, see the [Elastic Runtime 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 Elastic Runtime for Pivotal Cloud Foundry.
- **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 Elastic Runtime

 **Note:** If you are using the [Pivotal Network API](#), the latest product versions will automatically appear in your [Installation Dashboard](#).

To upgrade Elastic Runtime for PCF 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

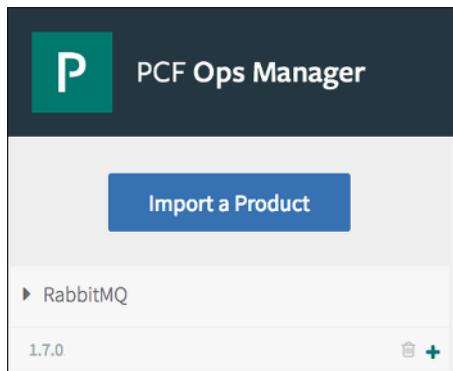
 **Note:** If you are using the [Pivotal Network API](#), the latest product versions will automatically appear in your [Installation Dashboard](#).

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 Elastic Runtime. 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 Elastic Runtime, 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, and vSphere, so 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 Cloud Foundry Elastic Runtime

Available Reference Architectures

- [Pivotal Cloud Foundry on AWS](#)
- [Pivotal Cloud Foundry on Azure](#)
- [Pivotal Cloud Foundry on GCP](#)
- [Pivotal Cloud Foundry on vSphere](#)

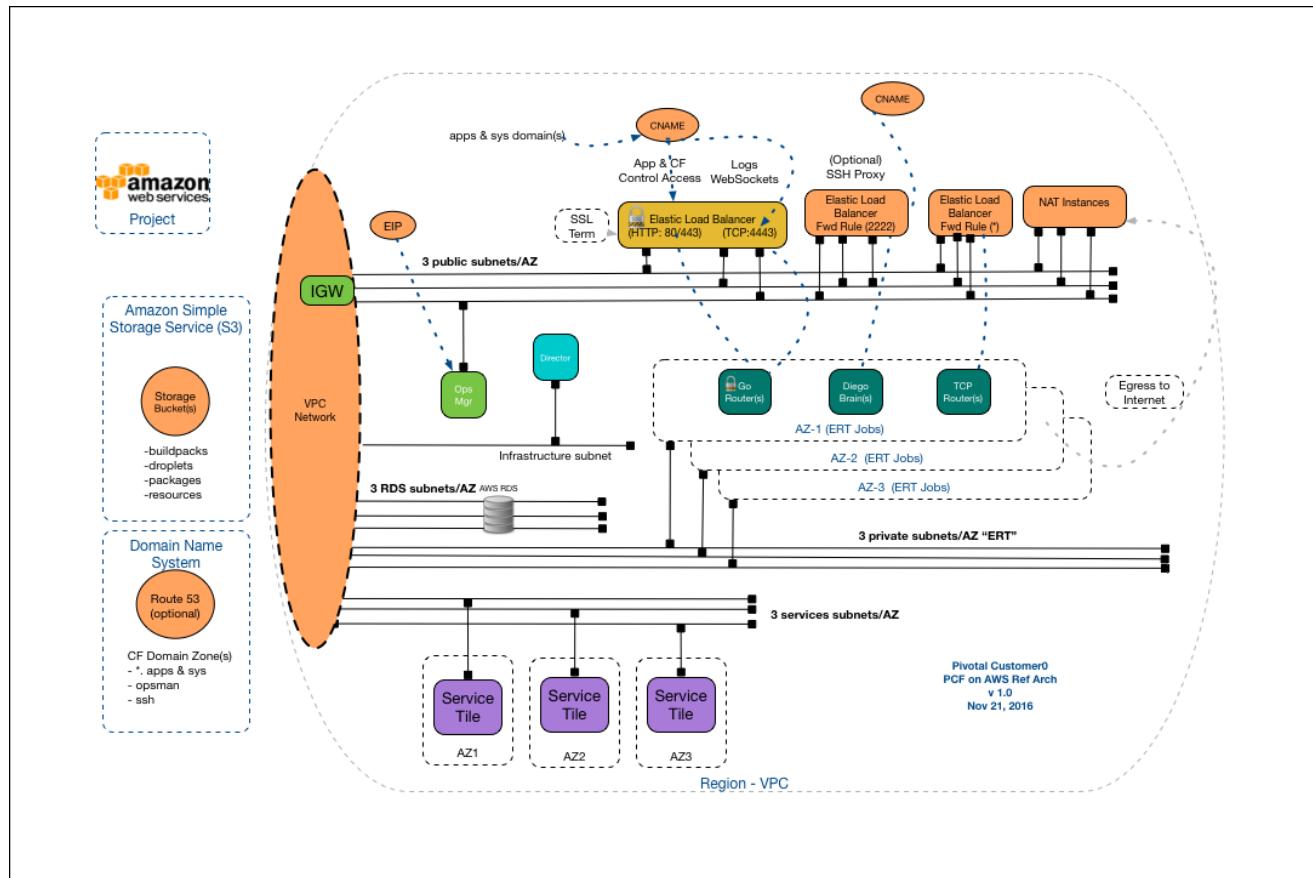
Reference Architecture for Pivotal Cloud Foundry on AWS

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).

Base Reference Architecture

The following diagram provides an overview of a reference architecture deployment of PCF on AWS using three AZs.



To view a larger version of this diagram, click [here](#).

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	<p>CF Domain Zones and routes in use by the reference architecture include:</p> <ul style="list-style-type: none"> • domains 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)

Ops Manager	Using Route 53 to manage domains is optional. Deployed on one of the three public subnets and accessible by FQDN or via 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 Gorouter(s). 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 via the HTTP, HTTPS, and SSL Elastic Load Balancers. Deployed on all three ERT subnets, one per AZ.
Diego Brains	This component is required. However, the SSH container access functionality is optional and enabled via the SSH Elastic Load Balancers. Deployed on all three ERT subnets, one per AZ.
TCP Routers	Optional feature for TCP routing. Deployed on all three ERT 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.
Service Accounts	<p>Two service accounts are recommended: one for Terraform, and the other for Ops Manager and BOSH. Consult the following list:</p> <ul style="list-style-type: none"> Admin Account: Terraform will use this account to provision required AWS resources as well as an IAM service account. IAM Service Account: This service account 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
Subnets	<p>The reference architecture requires the following subnets:</p> <ul style="list-style-type: none"> 1 x (/24) infrastructure (BOSH Director) subnet 3 x (/24) public subnets (Ops Manager, Elastic Load Balancers, NAT instances), one per AZ 3 x (/20) ERT 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 (/24) RDS subnets (Cloud Controller DB, UAA DB, etc.), one per AZ. <p>For more information, see the Terraform subnets script.</p>	13
Route Tables	<p>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:</p> <ul style="list-style-type: none"> PublicSubnetRouteTable: This routing table enables the ingress/egress routes from/to Internet through the Internet gateway for OpsManager and the NAT Gateway. PrivateSubnetRouteTable: This routing table enables the egress routing to the Internet through the NAT Gateway for the BOSH Director and ERT. <p>For more information, see the Terraform script that creates the route tables and the script that performs the route table association.</p>	4

Note: If an EC2 instance sits on a subnet with an Internet gateway attached as well as a public IP, it is accessible from the Internet through the public IP; for example, Ops Manager. ERT needs Internet access due to the access needs of using an S3 bucket as a blobstore.

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.

Security Group	Port	From CIDR	Protocol	Description
OpsMgrSG	22	0.0.0.0/0	TCP	Ops Manager SSH access
OpsMgrSG	443	0.0.0.0/0	TCP	Ops Manager HTTP access
VmsSG	ALL	VPC_CIDR	ALL	Open up connections among BOSH-deployed VMs
MysqlSG	3306	VPC_CIDR	TCP	Enable network access to RDS
ElbSG	80	0.0.0.0/0	TCP	HTTP to Elastic Runtime
ElbSG	443	0.0.0.0/0	TCP	HTTPS to Elastic Runtime
ElbSG	4443	0.0.0.0/0	TCP	WebSocket connection to Loggregator endpoint
SshElbSG	2222	0.0.0.0/0	TCP	SSH connection to containers

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 ([PcfElb](#)), the other to forward the traffic to the Diego Brain SSH proxy ([PcfSshElb](#)). For more information, see the Terraform load balancers [script](#).

The following table describes the required listeners for each load balancer:

ELB	Instance/Port	LB Port	Protocol	Description
PcfElb	gorouter/80	80	HTTP	Forward traffic to Gorouters
PcfElb	gorouter/80	443	HTTPS	SSL termination and forward traffic to Gorouters
PcfElb	gorouter/80	4443	SSL	SSL termination and forward traffic to Gorouters
PcfSshElb	diego-brain/2222	2222	TCP	Forward traffic to Diego Brain for container SSH connections

Each ELB binds with a health check to check the health of the back-end instances:

- [PcfElb](#) checks the health on Gorouter port 80 with TCP
- [PcfSshElb](#) checks the health on Diego Brain port 2222 with TCP

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 IaaS where Ops Manager does not have a public IP. In these cases, you can SSH into Ops Manager or any other component via the Jumpbox.

5

2

1

Reference Architecture for Pivotal Cloud Foundry on Azure

Page last updated:

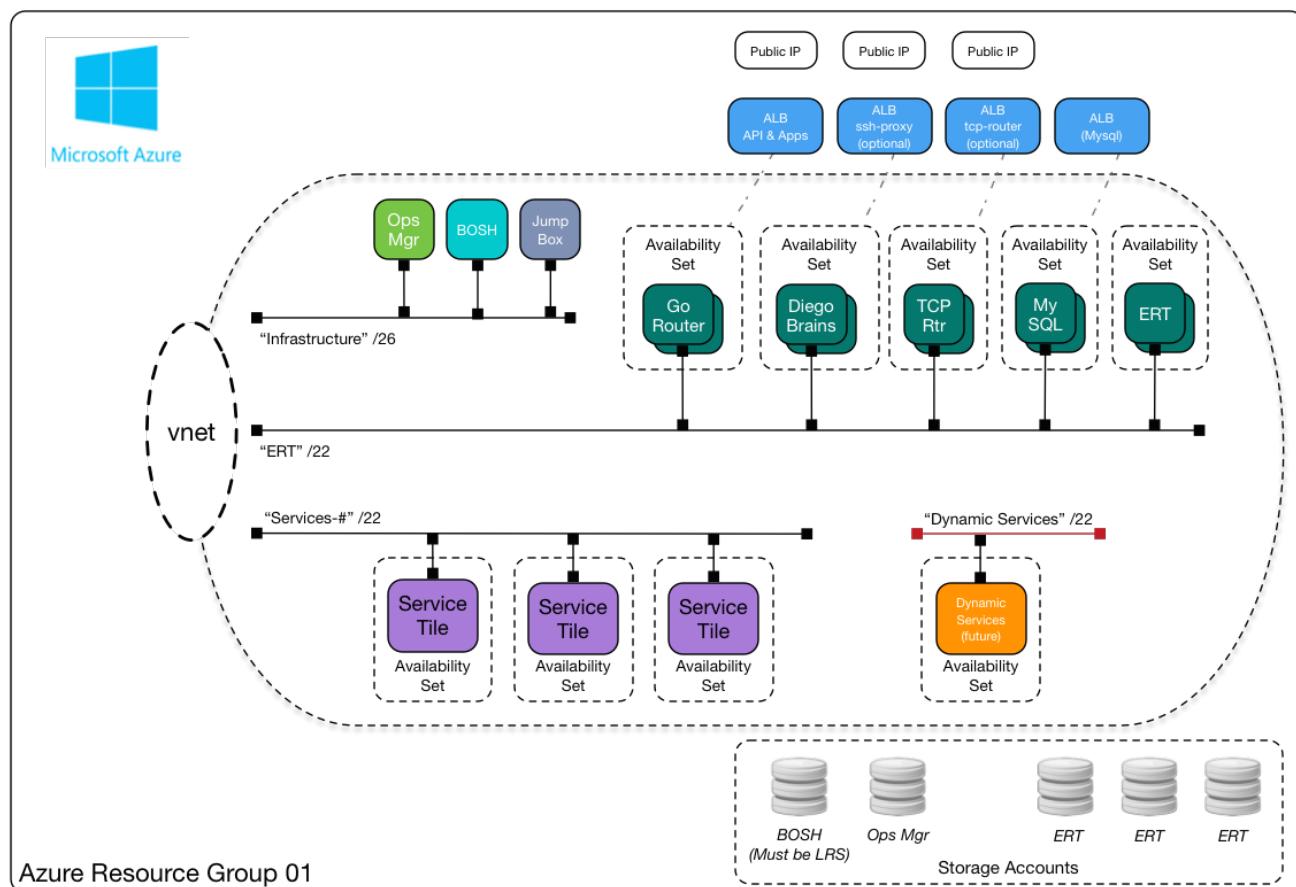
This guide presents a reference architecture for Pivotal Cloud Foundry (PCF) on Azure.

Azure does not provide resources in a way that translates directly to PCF availability zones. Instead, Azure provides high availability via 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.

Base Reference Architecture

The following diagram provides an overview of a reference architecture deployment of PCF on Azure.



To view a larger version of this diagram, click [here](#).

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 & DNS	<p>CF Domain Zones and routes in use by the reference architecture include:</p> <ul style="list-style-type: none"> domains for *.apps and *.system (required), a route for Ops Manager (required), a route for doppler (required), a route for loggregator (required),

	<ul style="list-style-type: none"> • 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 via an optional Jumpbox.
BOSH	Deployed on the infrastructure subnet.
Azure Load Balancer - API & Apps	Required. Load balancer that handles incoming API and apps requests and forwards them to the Gorouter(s).
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 Elastic Runtime.
Gorouter(s)	Accessed via the API & Apps load balancer. Deployed on the ERT subnet, one job per Azure availability set.
Diego Brain(s)	This component is required, however the SSH container access functionality is optional and enabled via the SSH Proxy load balancer. Deployed on the ERT subnet, one job per Azure availability set.
TCP Router(s)	Optional feature for TCP routing. Deployed on the ERT subnet, one job per availability zone.
MySQL	Reference architecture uses internal MySQL provided with PCF. Deployed on the ERT subnet, one job per Azure availability set.
Elastic Runtime	Required. Deployed on the ERT subnet, one job per Azure availability set.
Storage Accounts	PCF on Azure requires 5 standard storage accounts - BOSH, Ops Manager, and three ERT 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 (~20k, per each account), which generally relates to a BOSH JOB/VM limit of ~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.

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:

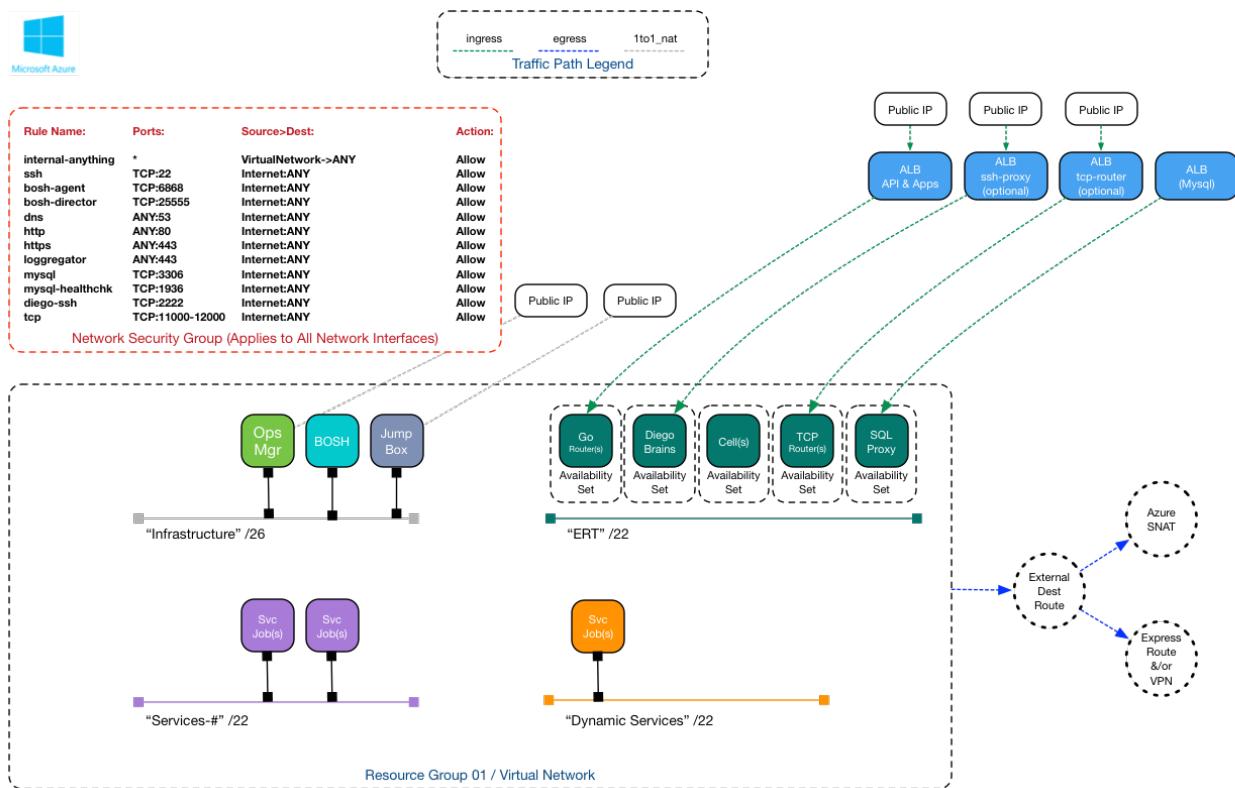
1. Single resource group, or
2. Multiple resource groups.

The first scenario is currently outlined in the [existing installation documentation](#) for Azure deployments of PCF. It models a single PCF deployment in a single Azure Resource Group.

If you require multiple resource groups, you may refer to the [Multiple Resource Group deployment](#) section.

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 IPs and deploy only one resource group.



To view a larger version of this diagram, click [here](#).

Network Objects

The following table lists the network objects in PCF on Azure reference architecture.

Network Object	Notes	Estimated Number
External Public IPs	Use <ol style="list-style-type: none"> global IP for apps and system access Ops Manager (or optional Jumpbox). Optionally, you can use a public IPs fro 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 <ol style="list-style-type: none"> infrastructure (Ops Manager, Ops Manager Director, Jumpbox), ERT, services, and dynamic services. Using separate subnets allows you to configure different firewall rules due to your needs.	4
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 & destination fields in a rule. One NSG can be used for all firewall rules.	12
Load	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	1-4

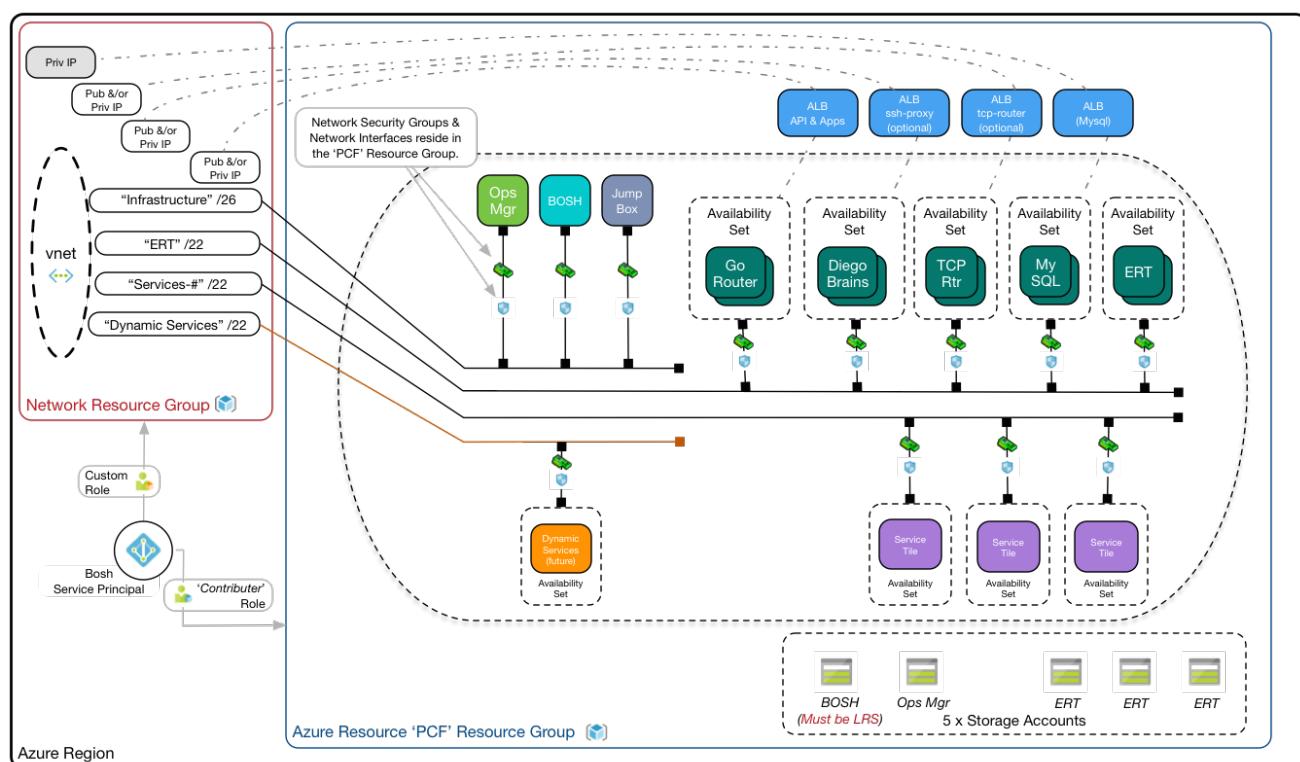
Balancers	load balancer that allows SSH access to Diego apps are both optional. In addition, you can use a MySQL load balancer to provide high availability to MySQL. This is also optional.	
Jumpbox	Optional. Provides a way of accessing different network components. For example, you can configure it with 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. In these cases, you can SSH into Ops Manager or any other component via 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.



To view a larger version of this diagram, click [here](#).

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/[YOURSUBSCRIPTIONID]"]
}
```

- 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]"]
}
```

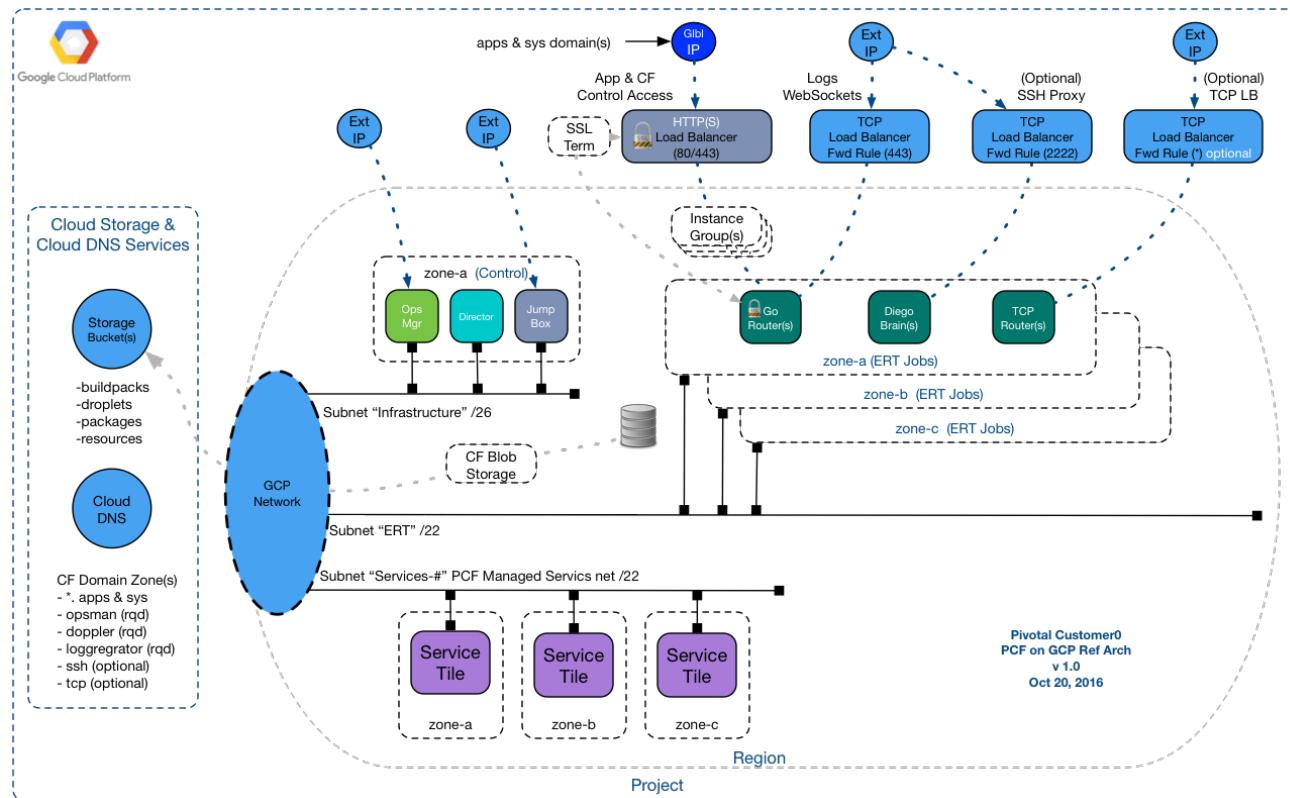
Reference Architecture for Pivotal Cloud Foundry on GCP

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.

Base Reference Architecture

The following diagram provides an overview of a reference architecture deployment of PCF on GCP.



To view a larger version of this diagram, click [here](#).

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
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 via an optional Jumpbox.
BOSH Director	Deployed on the infrastructure subnet.
Gorouter(s)	Accessed via the HTTP and TCP WebSockets load balancers. Deployed on the ERT subnet, one job per availability zone.
Diego Brain(s)	This component is required, however the SSH container access functionality is optional and enabled via the SSH Proxy load balancer. Deployed on the ERT subnet, one job per availability zone.
TCP Router(s)	Optional feature for TCP routing. Deployed on the ERT subnet, one job per availability zone.

Component	Reference Architecture Notes
Database	Reference architecture uses GCP Cloud SQL rather than internal databases.
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.

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:

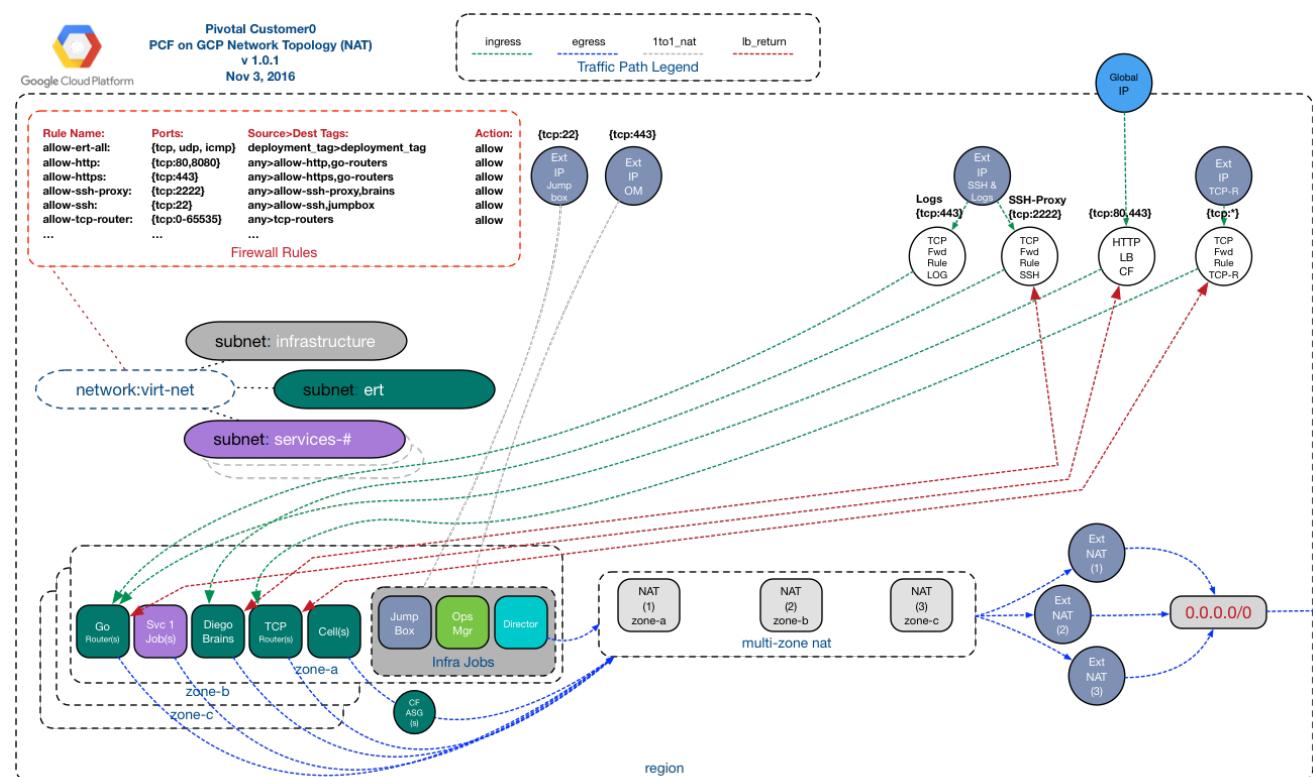
1. NATs provide public access to PCF internals, or
2. Every PCF VM receives its own public IP address (no NAT).

The latter scenario is currently outlined in the [existing installation documentation](#) for GCP deployments of PCF. Providing each PCF VM with a public IP 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 IPs.



To view a larger version of this diagram, click [here](#).

Public IPs 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 IPs
External IPs	For a NAT solution, use 1) global IP for apps and system access 2) 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 1) infrastructure (Ops Manager, Ops Manager Director, Jumpbox) 2) ERT 3) 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 2 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 HTTP load balancer provides SSL termination.	2+	2+
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 IaaS where Ops Manager does not have a public IP. In these cases, you can SSH into Ops Manager or any other component via the Jumpbox.	(1)	(1)

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 Architectures for Pivotal Cloud Foundry on vSphere

Page last updated:

This guide presents reference architectures for Pivotal Cloud Foundry (PCF) on vSphere.

Pivotal validates the reference architectures described in this topic against multiple production-grade usage scenarios. These test deployments host up to 1500 app instances and use PCF-managed services such as MySQL, RabbitMQ, and Spring Cloud Services.

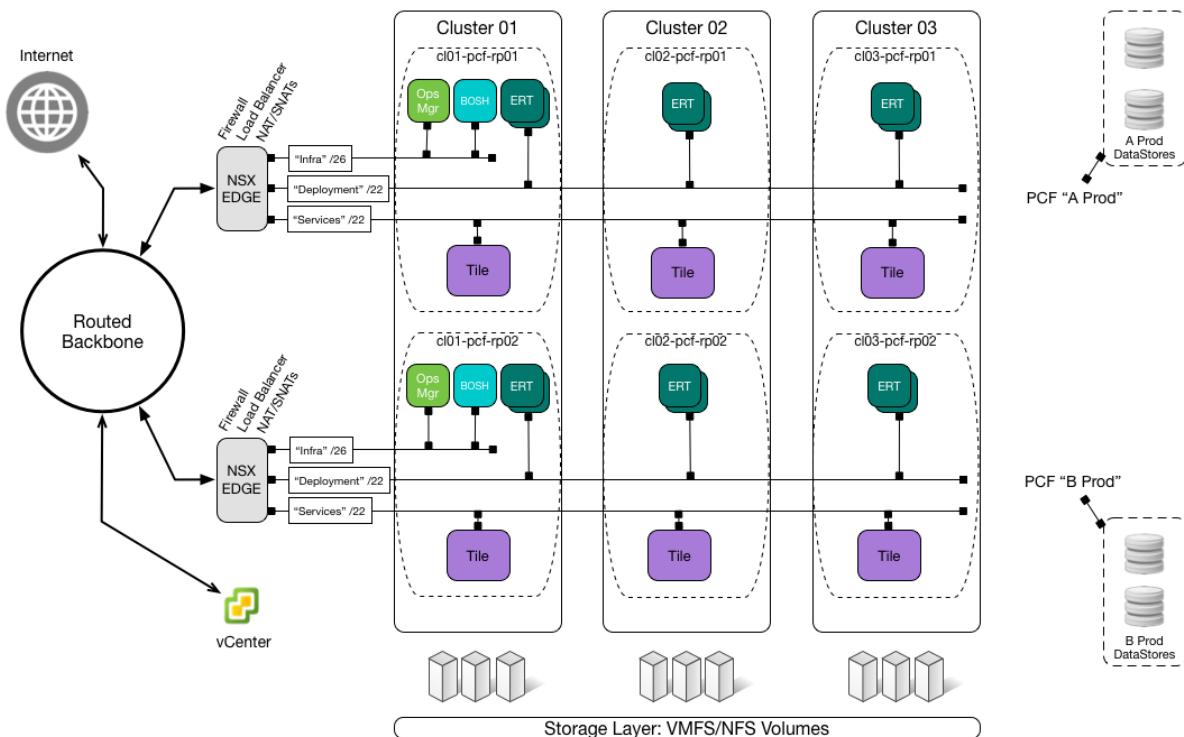
This document does not replace the [basic installation documentation](#), but gives proven examples of how to apply those instructions to real-world production environments.

PCF Products Validated	Version
PCF Ops Manager	1.8.latest
Elastic Runtime	1.8.latest

Base Reference Architecture

This recommended architecture relies on VMware NSX Edge, a software-defined services gateway that runs on VMware ESX/ESXi virtual hosts and combines a firewall, load balancer, and NAT/SNAT. See below for architectures that do not rely on NSX Edge.

The diagram below shows an architecture for two PCF installations sharing the same vSphere server clusters, yet segmented from each other with VMware Resource Pools. This design supports long term use, capacity growth at the vSphere level, and maximum installation security through the NSX Edge firewall. It allocates 3+ servers to each cluster, as recommended for vSphere, and spreads PCF components across 3 (or another odd number) of clusters, as [recommended](#) for PCF.



To view a larger version of this diagram, click [here](#).

Installation

To create a system following this architecture, do the following:

1. From vCenter, create three clusters. Pivotal recommends vSphere DVS (distributed virtual switching) for all clusters used by PCF.
2. Populate each cluster with two VMware Resource Pools. Enable VMware distributed resource scheduler (DRS) for each Resource Pool, so vMotion can automatically migrate data to avoid downtime.
3. For hosting capacity, populate each cluster with three ESXi hosts, making nine hosts for each installation. All installations collectively draw from the same nine hosts.
4. In one PCF deployment, use Ops Manager to create three Availability Zones (AZs), each corresponding to one of the Resource Pools from each cluster.
5. In the other PCF deployment, create an AZ for each of the three remaining Resource Pools.
6. For storage, add dedicated datastores to each PCF deployment following one of the two approaches, vertical or horizontal, as described [below](#).
7. Supply core networking for each deployment by configuring an NSX Edge with the following subnets. See[below](#) for details:
 - Infrastructure
 - Elastic Runtime (ERT)
 - Services

Scaling

You can easily scale up this architecture to support additional PCF installations with the same capacity, keeping each one resource-protected and separated.

To support more PCF installations, scale this architecture vertically by adding Resource Pools. To add capacity to all PCF installations, scale it horizontally by adding hosts to the existing clusters in sets of three, one per cluster.

Priority

In this architecture, multiple PCF installations share host resources. You can use vCenter resource allocation shares to assign [High](#), [Normal](#), or [Low](#) priority to pools used by different installations. 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 (for example) by assigning its resource pools a [High](#) share value setting.

Storage Configuration

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 6 datastores [ds01](#) through [ds06](#), you grant all nine hosts access to all six datastores, then provision PCF installation #1 to use stores [ds01](#) through [ds03](#), and installation #2 to use [ds04](#) through [ds06](#). Installation #1 will use [ds01](#) until it is full, then [ds02](#), and so on.
- **Vertical:** You grant each host cluster its own dedicated datastores, giving each installation multiple datastores based on their host cluster. vSphere VSAN storage requires this architecture. With 6 datastores [ds01](#) through [ds06](#), for example, you assign datastores [ds01](#) and [ds02](#) to cluster 1, [ds03](#) and [ds04](#) to cluster 2, and [ds05](#) and [ds06](#) to cluster 3. Then you provision PCF installation #1 to use [ds01](#), [ds03](#) and [ds05](#), and installation #2 to use [ds02](#), [ds04](#) and [ds06](#). With this arrangement, all VMs in the same installation and cluster share a dedicated datastore.

 **Note:** If a vSphere datastore is part of a vSphere Storage Cluster using sDRS (storage DRS), you must disable the sDRS feature on any datastores used by PCF. Otherwise, vMotion activity can rename independent disks and cause BOSH to malfunction.

Storage Capacity and Type

- **Capacity:** Pivotal recommends allocating at least 16TB of data storage for a typical PCF installation, either as two 8TB stores or a greater number of smaller volumes. Small installations without many tiles can use less; two 4TB volumes is reasonable.
- **Type:** Pivotal recommends block-based (fiber channel or iSCSI) and file-based (NFS) over high-speed carriers such as 6G FC or 10GigE. Redundant storage is highly recommended for any persistent data, but you can use DASD or JBOD for ephemeral data.

Networking

Using VMware NSX SDN (software-defined networking) provides the following benefits:

- Firewall capability per-installation through the built-in Edge firewall
- High capacity, resilient load balancing per-installation through the NSX Load Balancer
- Installation obfuscation through the use of non-routed RFC networks behind the NSX Edge and the use of SNAT/DNAT connections to expose only the endpoints of Cloud Foundry that need exposure
- High repeatability of installations through the repeat use of all network and addressing conventions on the right hand side of the diagram (the Tenant Side)
- Automatic rule and ACL sharing via NSX Manager Global Ruleset
- Automatic HA pairing of NSX Edges, managed by NSX Manager
- Support for PCF Go Router IP membership in the NSX Edge virtual load balancer pool by the BOSH CPI (not an Ops Manager feature)

Networking Design

Each PCF installation consumes three (or more) networks from the NSX Edge, aligned to specific job types:

- **Infrastructure:** This inward-facing network has a small CIDR range and 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.
- **Deployment:** Also known as the *apps wire*, this network has a large CIDR range. It hosts the Diego cell VMs that Elastic Runtime deploys apps into, and it also hosts Elastic Runtime support components.
- **Services:** This network (or multiple networks) has a large CIDR range. It hosts services that are installed with Ops Manager tiles and managed by BOSH.
PCF services are either pre-provisioned or [on-demand](#). The on-demand services require their own dedicated network, so an installation offering both types of services needs at least two services networks. A more involved approach would be to deploy multiple “Services-#” networks, one for each tile or each category of service function, for example databases, message buses, and so on.

All of these networks are considered “inside” or “tenant-side” networks, and use non-routable RFC network space to make provisioning repeatable. The NSX Edge translates between the tenant and service provider side networks using SNAT and DNAT.

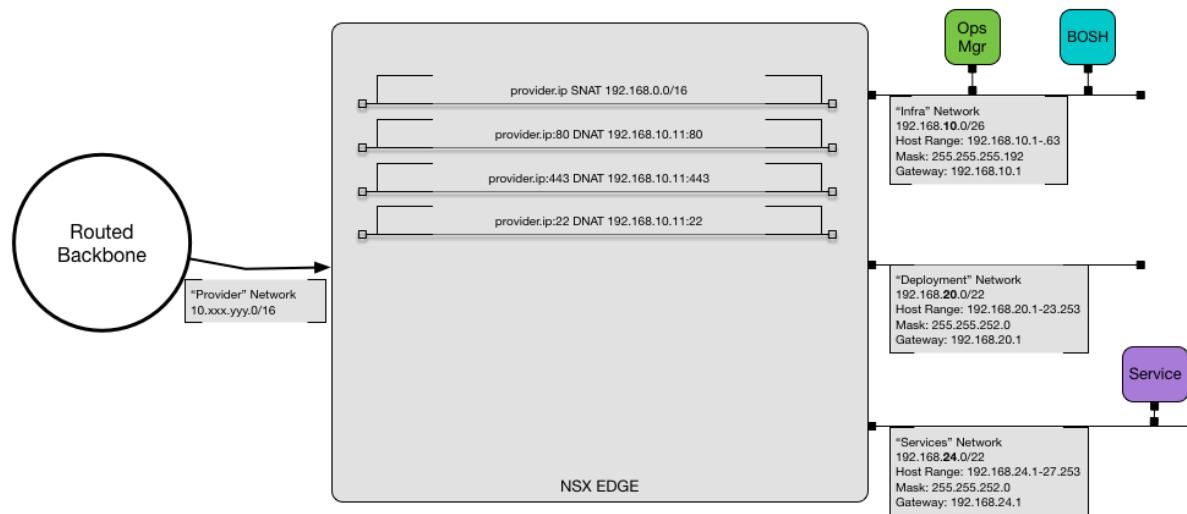
Provision each NSX Edge with at least four routable IP addresses from the service provider:

1. A static IP by which NSX Manager manages the NSX Edge
2. A static IP for use as egress SNAT (traffic from the tenant side exits the Edge on this IP)
3. A static IP for DNATs to Ops Manager
4. A static IP for the load balancer VIP that balances to a pool of PCF Gorouters

In addition to these four, there are many more uses for IPs on the routed side of the NSX Edge. Pivotal recommends reserving ten contiguous, static IPs per NSX Edge for future needs and flexibility.

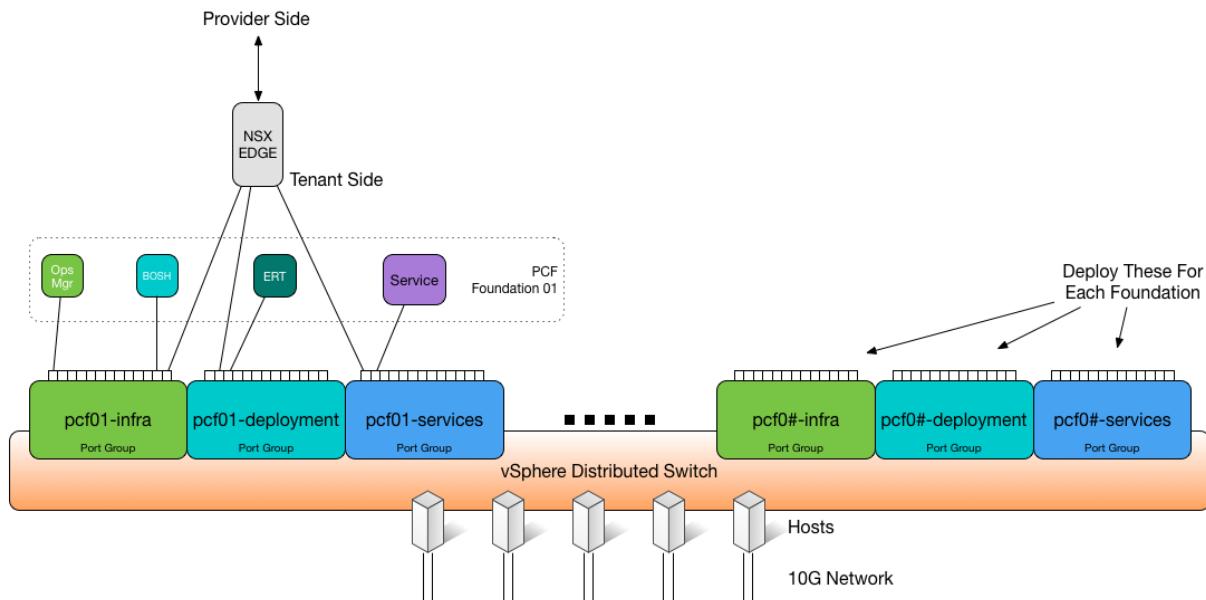
On the tenant side, each interface defined on the NSX Edge acts as the IP gateway for the network used on that port group. Pivotal recommends allocating the following address ranges for the networks, and defining the gateway at [.1](#) for each:

- **Infra** (infrastructure) network: 192.168.10.0/26
- **Deployment** network: 192.168.20.0/22
- **Services** network: 192.168.24.0/22
- **Services-B** network(s): 192.168.28.0/22, and so on...



To view a larger version of this diagram, click [here](#).

For each network interface provisioned on the NSX Edge, NSX creates a DPG (distributed port group).



To view a larger version of this diagram, click [here](#).

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.

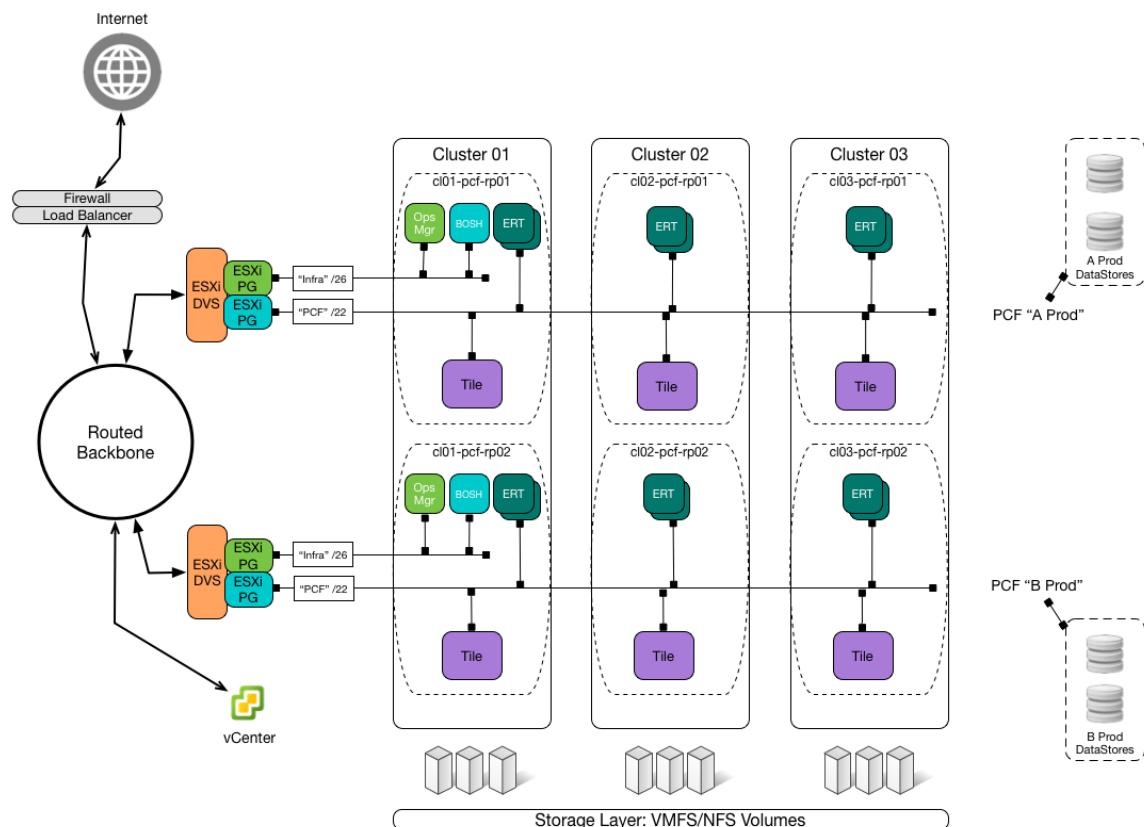
Networking Features

- Load balancing is handled by an external service, such as a hardware appliance or a VM from a 3rd party.
- An external service also performs SSL termination.

- You need to set up firewalls for each zone or network inside the installation, rather than having the NSX Edge appliance span multiple networks.
- To obfuscate network addresses, you need to configure a SNAT/DNAT and single or possibly multiple VLANs from the routable network, rather than turn on the SNAT/DNAT functionality of the NSX Edge.

Networking Design

The more traditional approach without SDN would be to deploy a single VLAN for use with all of PCF, or possibly a pair of VLANs, one for infrastructure and one for PCF.



To view a larger version of this diagram, click [here](#).

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:

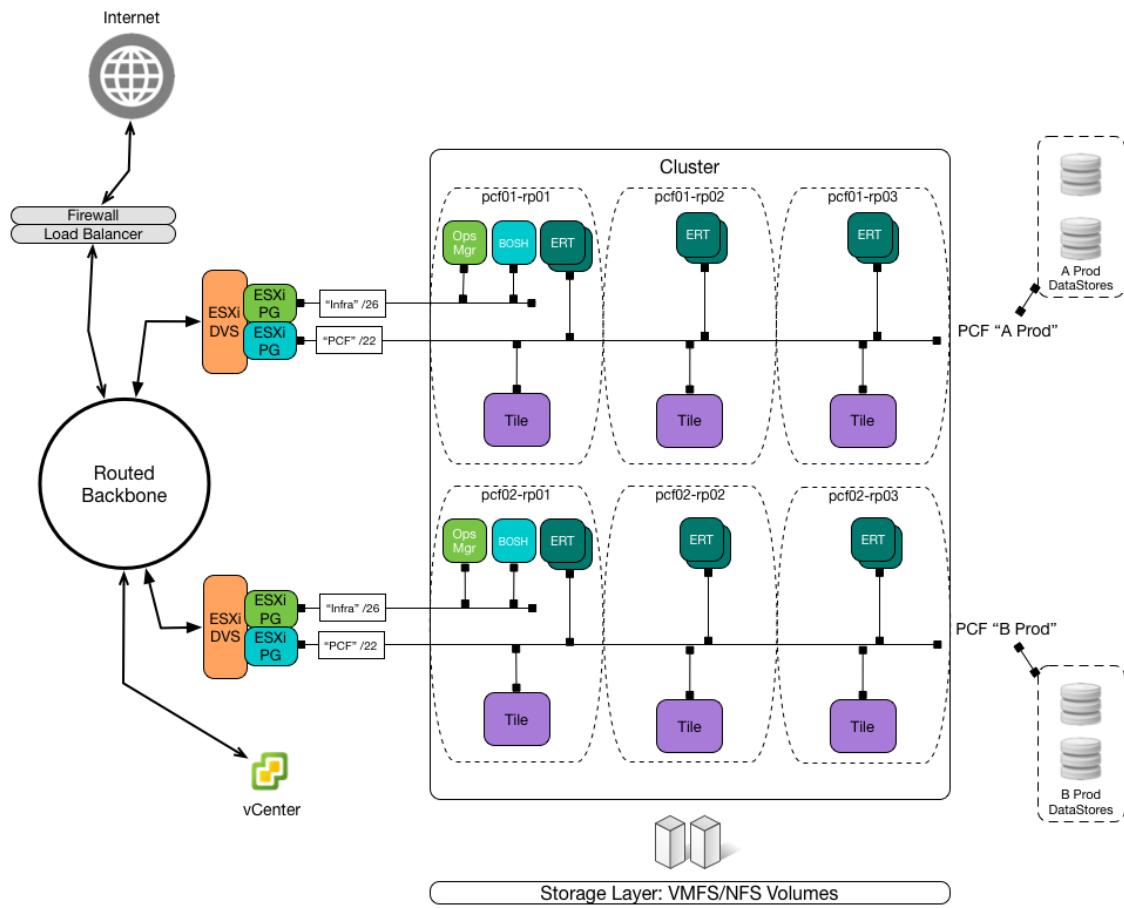
1. **Infra:** CPI, BOSH, and Ops Manager VMs that communicate with the IaaS layer
2. **PCF:** the deployment network for all tiles, including ERT

In a typical installation, you assign each of these port groups to a VLAN out of 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 (TOR) or end-of-row (EOR) switch/router appliance.

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.

To reduce resource use even further, you can place all hosts into a single cluster with VMware DRS and HA (high availability) enabled.



To view a larger version of this diagram, click [here](#).

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 would force the operator into aligning odd-numbered components into two AZs, which does not work well for PCF internal voting algorithms. If you do not want to consume three clusters for PCF, using one 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.

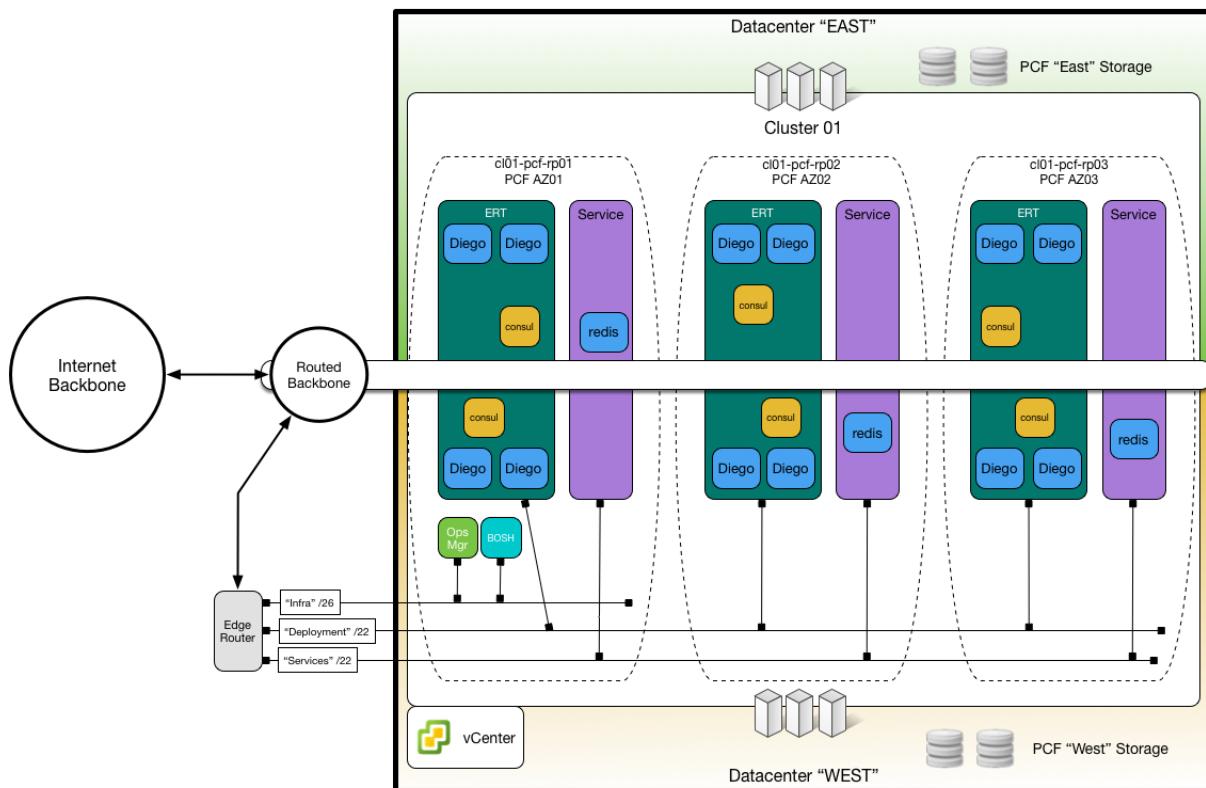
One interesting strategy for high availability is to keep a record of how many hosts are in a cluster and deploy enough copies of a PCF component in that AZ to ensure survivability in a site loss. This means placing large, odd numbers of components (such as consul) in the cluster so that at least two components are left on either site in the event of a site outage. In a four host cluster, this would call for five consul VMs, so each site has at least two if not the third. 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 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 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 you can use NSX or another SDN solution to tunnel one location over the other.



To view a larger version of this diagram, click [here](#).

PCF and BOSH treat the stretched cluster as an AZ, and make the same demands on it that they do with any other AZ. So 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 4x3x3 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 single or 2-core VMs can handle your apps, favor them over 4- and 8-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 three resource 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.

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 Elastic Runtime, 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	<code>cf dev start</code>	Ops Manager	<code>bosh deploy</code>
Estimated time-to-deploy	10 Minutes	Hour+	Hour+
Out-of-the-Box Services	Redis MySQL RabbitMQ	Redis MySQL RabbitMQ GemFire	N/A
Elastic Runtime	✓	✓	✓
Logging/Metrics	✓	✓	✓

	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	✓		

Using Ops Manager

 **Note:** Pivotal Cloud Foundry (PCF) for vCloud Air and vCloud Director is deprecated and availability is restricted to existing customers. Contact [Support](#) for more information.

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](#)
- [Understanding Floating Stemcells](#)
- [Configuring Ops Manager Director for AWS](#)
- [Configuring Amazon EBS Encryption](#)
- [Configuring Ops Manager Director for VMware vSphere](#)
- [Creating UAA Clients for BOSH Director](#)
- [Configuring Ops Manager Director for OpenStack](#)
- [Deploying Elastic Runtime on AWS](#)
- [Configuring Elastic Runtime for vSphere](#)
- [Installing Elastic Runtime after Deploying Pivotal Cloud Foundry on OpenStack](#)
- [Using Your Own Load Balancer](#)
- [Understanding Pivotal Cloud Foundry User Types](#)
- [Starting and Stopping Pivotal Cloud Foundry Virtual Machines](#)
- [Creating and Managing Ops Manager User Accounts](#)
- [Creating New Elastic Runtime 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](#)
- [Creating a Proxy ELB for Diego SSH without CloudFormation](#)

Monitoring, Logging, and Troubleshooting

- [Monitoring Virtual Machines in Pivotal Cloud Foundry](#)
- [Pivotal Cloud Foundry Troubleshooting Guide](#)
- [Troubleshooting Ops Manager for VMware vSphere](#)
- [Recovering MySQL from Elastic Runtime Downtime](#)
- [Advanced Troubleshooting with the BOSH CLI](#)

Using the Ops Manager API

This topic explains how to get started using the Ops Manager API. For the complete Ops Manager API documentation, browse to <https://YOUR-OPS-MANAGER-FQDN/docs>.

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 steps:

1. SSH into the Ops Manager VM:

```
$ ssh -i opsman ubuntu@OPS-MAN-FQDN
```

If the private key that you generated when deploying your Ops Manager Director instance is not named `opsman`, provide the correct filename instead.

2. From the Ops Manager VM, use the UAAC to target your Ops Manager UAA server:

```
$ uaac target https://OPS-MAN-FQDN/uaa
```

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
```

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.

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"
```

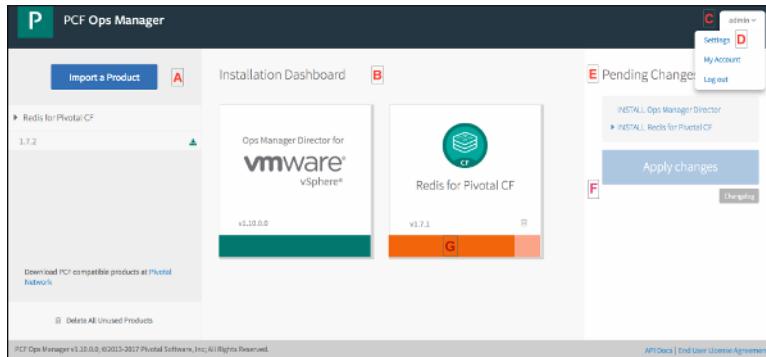
The request produces the following response:

```
[ {"installation_name": "p-bosh", "guid": "p-bosh-  
-00000000000000000000", "type": "p-  
bosh", "product_version": "1.10.0.  
0"}, {"installation_name": "cf-  
0000000000000000", "guid": "cf-0000000000000000  
000000", "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.

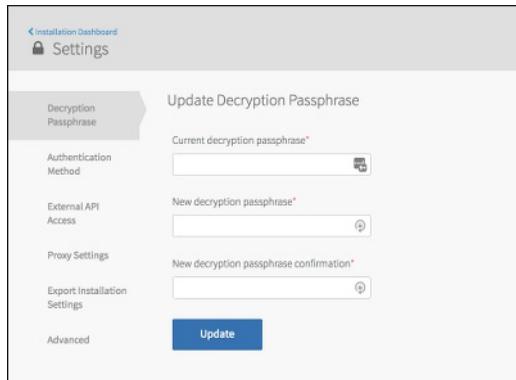


- A**— Displays a list of products you have imported that are ready for installation.
 - Click the **Import a Product** link to add a new product to Ops Manager.
 - If an upgrade is available, an active **Upgrade** button appears when you hover over the name of the product. If you are using the [Pivotal 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 on the **Change Log** button.
- 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**.

The screenshot shows the 'Account Settings' page. The 'Profile' section displays the email address 'admin@test.org' and a masked password. It includes links to 'Change Email' and 'Change Password'. The 'Third Party Access' section indicates that no third-party applications have been authorized yet.

Account Settings	
Profile	
admin@test.org *****	Change Email Change Password
Third Party Access	
You have not yet authorized any third party applications.	

Adding and Deleting Products

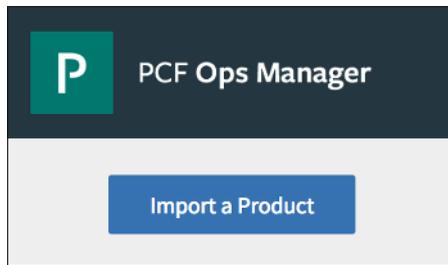
Page last updated:

Refer to this topic for help adding and deleting additional products from your [Pivotal Cloud Foundry](#) (PCF) installation, such as [Pivotal HD for PCF](#) and [Pivotal RabbitMQ® for PCF](#).

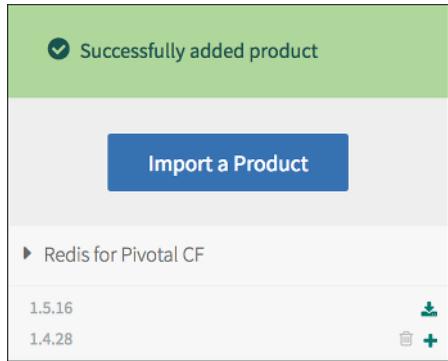
 **Note:** In Ops Manager 1.9, all product tiles use floating stemcells by default. This increases the security of your deployment by enabling tiles to automatically use the latest patched version of a stemcell, but it may significantly increase the amount of time required by a tile upgrade. Review the [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**.



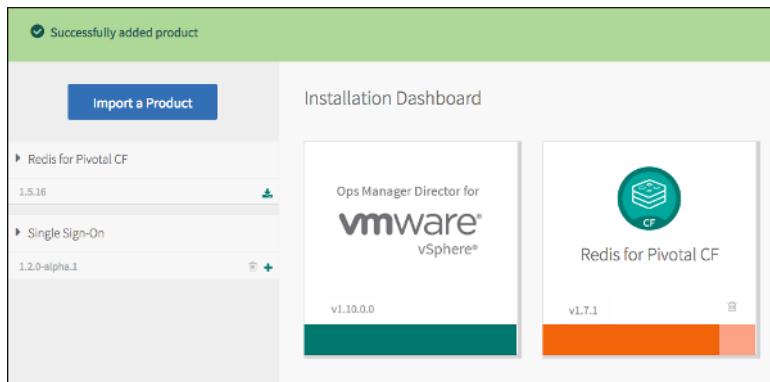
4. Select the .pivotal file that you downloaded from Pivotal Network or received from your software distributor, then click **Open**. If the product is successfully added, it appears in the your product list. If the product you selected is not the latest version, the most up-to-date version will appear 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 screenshot shows the Redis for Pivotal CF product configuration view. The left sidebar lists configuration tabs: Settings (selected), Status, Credentials, and Logs. The main area shows the 'Errands' tab selected. It contains three sections: 'Errands' (listing 'Assign AZs and Networks', 'Shared-VM Plan', 'Syslog', and 'Errands'), 'Post-Deploy Errands' (listing 'Broker Registrar' with a checked checkbox), and 'Pre-Delete Errands' (listing 'Broker Deregistrar' with a checked checkbox). A large blue 'Save' button is at the bottom.

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**.

The screenshot shows a rectangular form with a thin black border. Inside, there is a text input field containing the text "JW...". To the left of the input field is the text "API TOKEN". To the right is a blue "Regenerate" button.

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**.

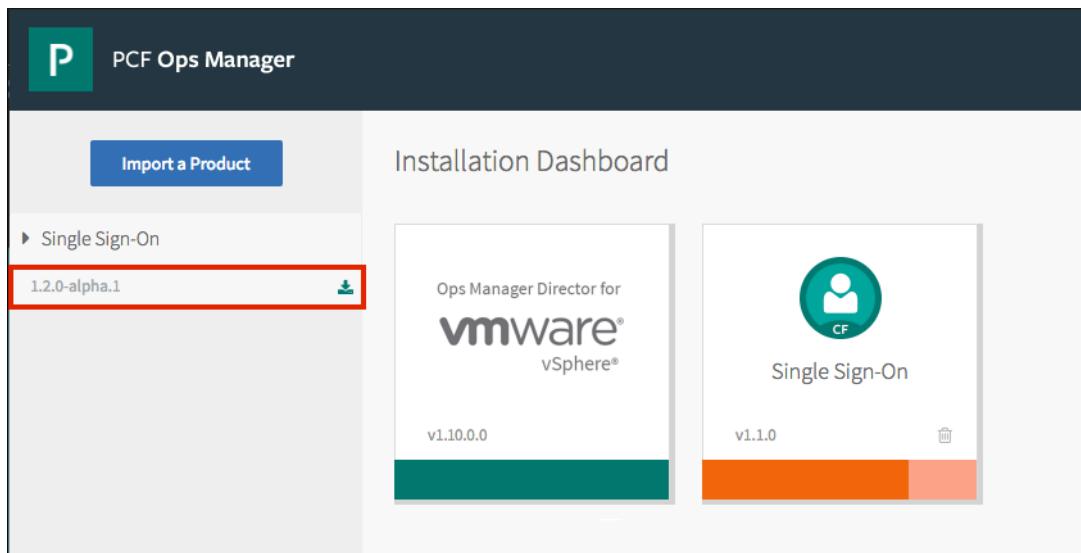
The screenshot shows a "Settings" page with a sidebar on the left containing links like "Decryption Passphrase", "Authentication Method", "External API Access" (which is highlighted in grey), "Proxy Settings", "Export Installation Settings", and "Advanced". The main content area has a title "Pivotal Network Settings" and a sub-instruction "Configure your [Pivotal Network](#) API token to enable update checking." Below this is a "Set API token" section with an input field containing "JW...", a "Save" button, and a "Cancel" button.

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**.

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.

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 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. Provide your SSH key, or when prompted, enter the password you configured for SSH access during Ops Manager deployment.

```
$ ssh ubuntu@OPS-MANAGER-FQDN  
Password: *****
```

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
```

- 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
```

- Target BOSH using the client. Replace `OPS-MANAGER-DIRECTOR-IP` with the IP address of your Ops Manager Director VM.

```
$ bosh --ca-cert /var/tempest/workspaces/default/root_ca_certificate \  
target OPS-MANAGER-DIRECTOR-IP
```

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.

- Select **Provision an admin client in the Bosh UAA** when configuring Ops Manager for SAML.
- After deploying Ops Manager Director (BOSH), click the Credentials tab in the Ops Manager Director tile.
- Click the link for the **Uaa Bosh Client Credentials** to get the client name and secret.
- Open a terminal and SSH into the Ops Manager VM. Provide your SSH key, or, when prompted, enter the password you configured for SSH access during Ops Manager deployment.

```
$ ssh ubuntu@OPS-MANAGER-FQDN  
Password: *****
```

- Set the client and secret as environment variables on the 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
```

- Target BOSH using the client. Replace `OPS-MANAGER-DIRECTOR-IP` with the IP address of your Ops Manager Director VM.

```
$ bosh --ca-cert /var/tempest/workspaces/default/root_ca_certificate \  
target OPS-MANAGER-DIRECTOR-IP
```

Using Your Own Load Balancer

Page last updated:

This guide describes how to use your own load balancer and forward traffic to your Elastic Runtime 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 IPs
- 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 Elastic Runtime.

Do not click **Install** after adding Elastic Runtime.

Step 4: Configure Networking

Complete the **Networking** configuration page in Elastic Runtime. Load balancer configuration in Elastic Runtime varies depending on which IaaS you are using for PCF. See the configuration procedure for your deployment IaaS— either [AWS](#), [Azure](#), [GCP](#), [OpenStack](#) or [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 Elastic Runtime installed to host apps, you add two more user types: [Elastic Runtime 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 Elastic Runtime 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 Elastic Runtime component VMs. The component VMs, in turn, support the VMs that host applications. Typical operator tasks include:

- Deploying and configuring Ops Manager, Elastic Runtime, and other product and service tiles.
- Maintaining and upgrading PCF deployments.
- Creating user accounts for Elastic Runtime users and the orgs that Elastic Runtime 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](#).

Elastic Runtime Users

Elastic Runtime users are app developers, managers, and auditors who work within orgs and spaces, the virtual compartments within a deployment where Elastic Runtime users can run apps and locally manage their roles and permissions.

A Role-Based Access Control (RBAC) system defines and maintains the different Elastic Runtime user roles:

- Org Manager, Org Auditor, Org Billing Manager
- Space Manager, Space Developer, Space Auditor

The [Orgs, Roles, Spaces, Permissions](#) topic describes the Elastic Runtime 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 Elastic Runtime users use system tools such as the cf CLI, PCF Metrics, and [Apps Manager](#), a dashboard for managing Elastic Runtime

users, orgs, spaces, and apps.

User Accounts

When an operator configures Elastic Runtime for the first time, they specify one of the following authentication systems for Elastic Runtime user accounts:

1. Internal authentication, using a new UAA database created for Elastic Runtime. 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, Elastic Runtime 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 Elastic Runtime users within their orgs and spaces. Elastic Runtime 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 Elastic Runtime 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 Elastic Runtime. They do not interact directly with Elastic Runtime components or interfaces. Any interactions or roles they perform within the apps are defined by the apps themselves, not Elastic Runtime.

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 Elastic Runtime user store, along with Elastic Runtime 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)	<ul style="list-style-type: none"> • IaaS UI • PivNet • Ops Manager • cf CLI • UAA CLI (UAAC) • SSO Dashboard • Marketplace 	Ops Manager user store through UAA <i>or</i> External store through SAML	Operators and Elastic Runtime Users
	<ul style="list-style-type: none"> • UAA Administrator • Org Manager • Org Auditor 	<ul style="list-style-type: none"> • cf CLI • CAPI 	Elastic Runtime user store	Elastic Runtime Users

Elastic Runtime Users	<ul style="list-style-type: none"> • Org Billing Manager • Space Manager • Space Developer • Space Auditor 	<ul style="list-style-type: none"> • Apps Manager • PCF Metrics • Marketplace 	through UAA <i>or</i> External store through SAML or LDAP	within permitted orgs and spaces, and End Users
End Users	Defined by apps they use	Hosted apps	Individual apps <i>or</i> Elastic Runtime user store through SSO	

Starting and Stopping Pivotal Cloud Foundry Virtual Machines

Page last updated:

This topic describes starting and stopping the component virtual machines (VMs) that make up a [Pivotal Cloud Foundry](#) (PCF) deployment. You may in some cases want to stop or start all of your PCF VMs, for instance to power down your deployment or to recover from a power outage. You can do this with a single command, or you can perform the process manually. If you want to shut down 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 [Prepare to Use the BOSH CLI](#) for help setting up this tool.

Start and Stop Your PCF VMs

This section describes how to start or stop all the VMs in your deployment with a single command.

Start

Run the following command to start all the VMs in your deployment:

```
$ bosh -d PATH-TO-CF-DEPLOYMENT start
```

Stop

Run the following command to shut down all the VMs in your deployment:

```
$ bosh -d PATH-TO-CF-DEPLOYMENT stop --hard
```

Start and Stop Your PCF VMs Manually

This section describes how to start and stop all the VMs in your deployment individually. Dependencies between the components in your PCF deployment require that you start and stop the VMs for those components in a specific order. These orders are specified below in the [start order](#) and [stop order](#) tables.

Find the Names for Your PCF Virtual Machines

You need the full names for the VMs to [start](#) or [stop](#) them using the BOSH CLI. To find full names for the VMs running each component, run

```
bosh vms :
```

```
$ bosh vms
Acting as user 'director' on 'p-bosh-399383d4525762cdc35c'
Deployment `cf-1ef2da789c0ed8f3567f

Director task 26

Task 26 done

+-----+-----+-----+
| VM | State | AZ | VM Type | IPs | IP
+-----+-----+-----+
| clock_global-partition-bb35e96d6d3184a2d672/0 (92174545-ea16-448c-bea8-5a3d27ef2078) | running | n/a | clock_global-partition-bb35e96d6d3184a2d672 | 192.0.2.101 |
| cloud_controller-partition-bb35e96dd3184a2d672/0 (a315eb23-04bf-4228-a346-0ff8cc936a86) | running | n/a | cloud_controller-partition-bb35e96d6d3184a2d672 | 192.0.2.100 |
| cloud_controller_worker-partition-bb35e96d6d3184a2d672/0 (62ccf688-29dc-42f4-9191-78cf6c363af) | running | n/a | cloud_controller_worker-partition-bb35e96d6d3184a2d672 | 192.0.2.105 |
| consul_server-partition-bb35e96d6d3184a2d672/0 (5eb05f9b-90d3-437f-aeeb-4a1b30320c77) | running | n/a | consul_server-partition-bb35e96d6d3184a2d672 | 192.0.2.92 |
| diego_brain-partition-bb35e96d6d3184a2d672/0 (2abe95cd-5094-4f87-bcb9-c9fecf8c6033) | running | n/a | diego_brain-partition-bb35e96d6d3184a2d672 | 192.0.2.104 |
| diego_cell-partition-bb35e96d6d3184a2d672/0 (23d12fad-ca7a-4e4a-9cd7-dc7d242e89ae) | running | n/a | diego_cell-partition-bb35e96d6d3184a2d672 | 192.0.2.105 |
| diego_cell-partition-bb35e96d6d3184a2d672/1 (9f94e756-f648-4c4b-a7e7-08099bd75263) | running | n/a | diego_cell-partition-bb35e96d6d3184a2d672 | 192.0.2.106 |
| diego_database-partition-bb35e96d6d3184a2d672/0 (0cca205b-bc68-42d5-95f0-47fc0c072bb6) | running | n/a | diego_database-partition-bb35e96d6d3184a2d672 | 192.0.2.95 |
| doppler-partition-bb35e96d6d3184a2d672/0 (a5bcd2ed-7b3c-4ebb-901c-614c2064d10c) | running | n/a | doppler-partition-bb35e96d6d3184a2d672 | 192.0.2.108 |
| etcd_server-partition-bb35e96d6d3184a2d672/0 (3e3b53cd-0b68-4f94-81cf-61da48dd20ab) | running | n/a | etcd_server-partition-bb35e96d6d3184a2d672 | 192.0.2.94 |
| ha_proxy-partition-bb35e96d6d3184a2d672/0 (bc7dad2-8e31-4d70-b314-f85eb51a503) | running | n/a | ha_proxy-partition-bb35e96d6d3184a2d672 | 192.0.2.254 |
| loggregator_trafficcontroller-partition-bb35e96d6d3184a2d672/0 (57157a16-6309-49f4-8133-b831b52bb363) | running | n/a | loggregator_trafficcontroller-partition-bb35e96d6d3184a2d672 | 192.0.2.96 |
| mysql_partition-bb35e96d6d3184a2d672/0 (ff18bb42-3df8-44b3-ba69-0b7ef12dfc30) | running | n/a | mysql_partition-bb35e96d6d3184a2d672 | 192.0.2.99 |
| mysql_proxy-partition-bb35e96d6d3184a2d672/0 (e4f72312-336d-465d-a8d0-57dc6b7abd66) | running | n/a | mysql_proxy-partition-bb35e96d6d3184a2d672 | 192.0.2.98 |
| nats_partition-bb35e96d6d3184a2d672/0 (654b2470-c6fb-40d1-a7d2-c7839a7c6403) | running | n/a | nats_partition-bb35e96d6d3184a2d672 | 192.0.2.93 |
| nfs_server-partition-bb35e96d6d3184a2d672/0 (df18f438-a0f1-46b2-b3ea-31bf96d47a18) | running | n/a | nfs_server-partition-bb35e96d6d3184a2d672 | 192.0.2.96 |
| router-partition-bb35e96d6d3184a2d672/0 (4fbcc68c-8bc6-425c-a4fc-04b7127b1f4a) | running | n/a | router-partition-bb35e96d6d3184a2d672 | 192.0.2.97 |
| uaa-partition-bb35e96d6d3184a2d672/0 (82c39142-da6e-42ec-af1c-b0247f74e8bd) | running | n/a | uaa-partition-bb35e96d6d3184a2d672 | 192.0.2.103 |
+-----+
VMs total: 17
```

You can see the full name of each VM in the `Job/index` column of the terminal output. Each full name includes:

- A prefix indicating the component function of the VM. The [table](#) below associates each component VM function with a prefix.
- The word `partition`
- An identifier string specific to your deployment
- An `/INDEX` suffix. For component processes that run on a single VM instance, INDEX is always 0. For processes running on multiple VMs, INDEX is a sequentially numbered value that uniquely identifies each VM.

For any component, you can look for its prefix in the `bosh vms` output to find the full name of the VM or VMs that run it. In the example shown here, the full name of one of the three Diego Cell VMs is `diego_cell-partition-458f9d7042365ff810e9/2`.

Start Your PCF Virtual Machines

In the order specified in the [Start Order table](#) below, run `bosh start VM-NAME` for each component in your PCF deployment. Use the full name of the component VM as listed in your `bosh vms` [terminal output](#), without the `/INDEX` at the end. In the example here, the first component you would start is the NATS VM, by running `bosh start nats-partition-458f9d7042365ff810e9`:

```
$ bosh start nats-partition-458f9d7042365ff810e9
Processing deployment manifest
-----
You are about to start nats-partition-458f9d7042365ff810e9/0
Detecting deployment changes
-----
Start nats-partition-458f9d7042365ff810e9/0? (type 'yes' to continue): yes
Performing `start nats-partition-458f9d7042365ff810e9/0'...
...
Started updating job nats-partition-458f9d7042365ff810e9 > nats-partition-458f9d7042365ff810e9/0 (canary). Done (00:00:43)
Task 42 done
nats-partition-458f9d7042365ff810e9/0 has been started
```

Note: To start a specific instance of a VM, include the `/INDEX` at the end of its full name. In the example here, you could start only the first Diego Cell instance by running:

```
$ bosh start diego_cell-partition-458f9d7042365ff810e9/0
```

Start Order	Component	Job/index name prefix (in <code>bosh vms</code> output)
1	NATS	<code>nats-</code>
2	consul	<code>consul_server-</code>
3	etcd	<code>etcd_server-</code>
4	Diego Database	<code>diego_database-</code>
5	WebDAV Server	<code>nfs_server-</code> (The WebDAV job has this prefix for historical reasons)
6	Router	<code>router-</code>
7	MySQL Proxy	<code>mysql_proxy-</code>
8	MySQL Server	<code>mysql-</code>
9	Cloud Controller Database	<code>ccdb-</code>
10	UAA Database	<code>uaadb-</code>
11	Cloud Controller	<code>cloud_controller-</code>
12	HAProxy	<code>ha_proxy-</code>
13	Health Manager	<code>health_manager-</code>
14	Clock Global	<code>clock_global-</code>
15	Cloud Controller Worker	<code>cloud_controller_worker-</code>
16	UAA	<code>uaa-</code>
17	Diego Brain	<code>diego_brain-</code>
18	Diego Cell	<code>diego_cell-</code>
19	Doppler Server	<code>doppler-</code>
20	Loggregator Traffic Controller	<code>loggregator_trafficcontroller-</code>

Stop Your PCF Virtual Machines

In the order specified in the [Stop Order table](#) below, run `bosh stop VM-NAME` for each component in your PCF deployment. Use the full name of the component VM as listed in your `bosh vms` [terminal output](#), without the `/INDEX` at the end. In the example here, the first component you would stop is the Loggregator Traffic Controller VM, by running `bosh stop loggregator_trafficcontroller-partition-458f9d7042365ff810e9`:

```
$ bosh stop loggregator_trafficcontroller-partition-458f9d7042365ff810e9

Processing deployment manifest
-----
You are about to stop loggregator_trafficcontroller-partition-458f9d7042365ff810e9/0

Detecting deployment changes
-----
Stop loggregator_trafficcontroller-partition-458f9d7042365ff810e9/0? (type 'yes' to continue): yes

Performing `stop loggregator_trafficcontroller-partition-458f9d7042365ff810e9/0'...

...
Started updating job loggregator_trafficcontroller-partition-458f9d7042365ff810e9 > loggregator_trafficcontroller-partition-458f9d7042365ff810e9/0 (canary). Done (00:00:37)

loggregator_trafficcontroller-partition-458f9d7042365ff810e9/0 has been stopped
```

Note: To stop a specific instance of a VM, include the `/INDEX` at the end of its full name. In the example here, you could stop only the third Diego Cell instance by running:

```
$ bosh stop diego_cell-partition-458f9d7042365ff810e9/2
```

Stop Order	Component	Job/index name prefix (in bosh vms output)
1	Loggregator Traffic Controller	loggregator_trafficcontroller-
2	Doppler Server	doppler-
3	Diego Cell	diego_cell-
4	Diego Brain	diego_brain-
5	UAA	uaa-
6	Cloud Controller Worker	cloud_controller_worker-
7	Clock Global	clock_global-
8	Health Manager	health_manager-
9	HAProxy	ha_proxy-
10	Cloud Controller	cloud_controller-
11	UAA Database	uaadb-
12	Cloud Controller Database	ccdb-
13	MySQL Server	mysql-
14	MySQL Proxy	mysql_proxy-
15	Router	router-
16	WebDAV Server	nfs_server- (The WebDAV job has this prefix for historical reasons)
17	Diego Database	diego_database-
18	etcd	etcd_server-
19	consul	consul_server-
20	NATS	nats-

Creating and Managing Ops Manager User Accounts

Page last updated:

Pivotal Cloud Foundry [\[x\]](#) 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 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
```


Creating New Elastic Runtime User Accounts

Page last updated:

When you first deploy your [Elastic Runtime](#) PaaS, 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 [Elastic Runtime > 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:

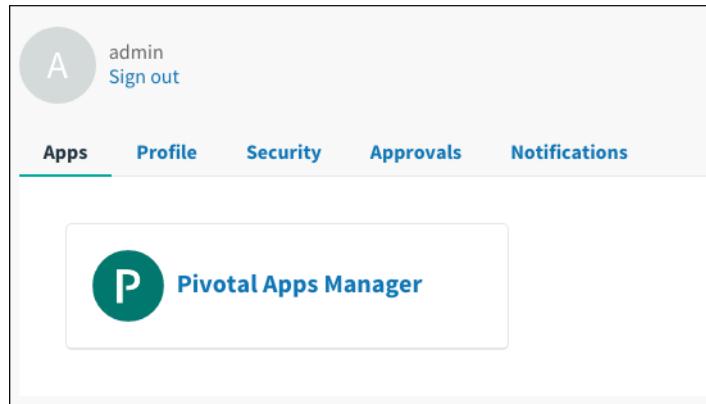
1. If you do not know the system domain for the deployment, then select **Pivotal Elastic Runtime > Settings > Domains** to locate the configured system domain.
2. Open a browser and navigate to `apps.YOUR-SYSTEM-DOMAIN`. For example, if the system domain is `system.example.com`, then point your browser 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 **Pivotal Elastic Runtime > Credentials > UAA > Admin Credentials**.

UAA	VM Credentials	Link to Credential
Admin Credentials		Link to Credential

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 Pivotal Cloud Foundry Deployment

Page last updated:

This topic describes the procedure for adding existing SAML or LDAP users to a [Pivotal Cloud Foundry](#) (PCF) deployment enabled with SAML or LDAP.

 **Note:** You must have admin access to the PCF Ops Manager Installation Dashboard for your deployment to complete the procedure described in this topic.

Step 1: Add SAML or LDAP Users

 **Note:** Do not create new users in Elastic Runtime using the Cloud Foundry Command Line Interface (cf CLI), by UAAC, or by using invitations in Apps Manager. Doing so creates a user identity in the internal user store, separate from the SAML or LDAP user identity. Instead, follow the procedure described below.

Two ways exist to add existing SAML or LDAP users to your PCF deployment:

- In bulk, using the UAA Bulk Import Tool. See [the UAA Users Import README](#) for instructions on installing and using the tool.
- Individually, through the cf CLI, as described below:
 1. Each existing SAML or LDAP user must log in to Apps Manager or to the cf CLI using their SAML (by entering `cf login --sso`) or LDAP credentials. Users will not have access to any org or space until these are granted by an Org or Space Manager.
 2. The PCF Admin must log in to the cf CLI and associate the user with the desired org and space roles. See [Org and App Space Roles](#).

(Advanced Option) Integrate with Enterprise Identity Management System

If your organization uses an Enterprise Identity Management System for centralized provisioning and deprovisioning of users, you can use the [Users API](#) and [Organizations API](#) to write a connector to manage users and permissions in Elastic Runtime.

Step 2: Create User

1. Run the command below to create the user in UAA. Replace 'EXAMPLE-USERNAME' with the username of the SAML or LDAP user you wish to add.

- For LDAP, set user `origin` to `ldap`.

```
$ uaac curl -H "Content-Type: application/json" -k /Users -X POST -d '{"userName":"EXAMPLE-USERNAME", "emails":[{"value":"EXAMPLE-USERNAME@test.com"}], "ori...
```

- For SAML, set user `origin` to the SAML identity provider name [set in](#) the Elastic Runtime tile under **Authentication and Enterprise SSO**.

```
$ uaac curl -H "Content-Type: application/json" -k /Users -X POST -d '{"userName":"EXAMPLE-USERNAME", "emails":[{"value":"EXAMPLE-USERNAME@test.com"}], "ori...
```

2. Use the [Users API](#) to create a User record in the Cloud Controller Database with the existing user's SAML or LDAP GUID.

```
$ curl "https://api.YOUR-DOMAIN/v2/users" -d '{  
  "guid": "YOUR-USER-GUID"  
}' -X POST \  
-H "Authorization: bearer YOUR-BEARER-TOKEN" \  
-H "Host: YOUR-HOST-URL" \  
-H "Content-Type: application/x-www-form-urlencoded" \  
-H "Cookie: "
```

Step 3: Provide User Access to Orgs

Use the [Organizations API](#) to associate the user with the appropriate orgs in your Elastic Runtime deployment.

Step 4: Associate User with Space or Org Role

You can grant Space and Org roles to users using the following API calls:

- [Associate an Auditor with a Space](#)
- [Associate a Developer with a Space](#)
- [Associate a Manager with a Space](#)
- [Associate an Auditor with a Organization](#)
- [Associate a Manager with a Organization](#)

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 cleanup task that removes all data objects used by the errand.

Elastic Runtime provides several post-deploy errands including smoke tests, Apps Manager, notification, Pivotal Account, and autoscaling errands. For more information about Elastic Runtime errands, see the *Deploying Elastic Runtime* topic for the platform where you are deploying Elastic Runtime. For example, if you are deploying PCF on GCP, see [Deploying Elastic Runtime 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 associated product manifest has changed since the last time the errand succeeded. If there are no changes to 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 Elastic Runtime 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 Ops Manager's default for the errand or any tile-specific default that overrides it.

Errands

Errands are scripts that run at designated points during an installation.

Post-Deploy Errands

Smoke Test Errand	Runs Smoke Tests against your Elastic Runtime installation
On	▼
Apps Manager Errand	Pushes the Pivotal Apps Manager application to your Elastic Runtime installation
On	▼
Notifications Errand	Pushes the Pivotal Notifications application to your Elastic Runtime installation
On	▼
Notifications UI Errand	Pushes the Notifications UI component to your Elastic Runtime installation
On	▼
Pivotal Account Errand	Pushes the Pivotal Account application to your Elastic Runtime installation
On	▼
Autoscaling Errand	Pushes the Pivotal App Autoscaling application to your Elastic Runtime installation
On	▼
Autoscaling Registration Errand	Registers the Autoscaling Service Broker
On	▼
NFS Broker Errand	Pushes the NFS Broker application to your Elastic Runtime installation
On	▼

There are no pre-delete errands for this product.

Save

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 dropdowns 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.

The screenshot shows the 'Pending Changes' section of the Ops Manager interface. At the top, there's a 'Revert' button. Below it, a list of errands for the 'UPDATE Pivotal Elastic Runtime' product is displayed. Each errand has a dropdown menu next to its name. The errands listed are: Smoke Test Errand (When Changed), Apps Manager Errand (On), Notifications Errand (On), Notifications UI Errand (On), Pivotal Account Errand (On), Autoscaling Errand (On), Autoscaling Registration (On), and NFS Broker Errand (Off). At the bottom of the list is a large blue 'Apply changes' button, and below it is a 'Changelog' link.

1. Use the dropdowns 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.
2. 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 Developers Guide*.

Backing Up and Restoring Pivotal Cloud Foundry

Page last updated:

The following considerations are important when backing up data in your Pivotal Cloud Foundry (PCF) deployment:

- If your deployment uses external databases, for example, AWS RDS, you must back up your data according to the instructions provided by your database vendor.
- If your PCF deployment originated from 1.5.x or earlier, follow the backup/restore instructions for PostgreSQL databases and for the MySQL server.
- If your PCF deployment originated from 1.6.0 or later, follow the backup/restore instructions for the MySQL server, but not those for PostgreSQL databases.
- If you do not know the original version of your PCF deployment, perform the following to determine what databases your deployment uses:
 1. From the Ops Manager Installation Dashboard, click **Pivotal Elastic Runtime**.
 2. Click **Databases** to determine whether your deployment uses internal databases with MySQL and PostgresQL, internal databases with

Choose the location of your system databases*

Internal Databases - MySQL and Postgres (the Postgres DBs are not highly-available, but this selection is required if you want to keep your system data from a pre-1.6.0 Elastic Runtime that you upgraded)

Internal Databases - MySQL (preferred for complete high-availability)

External Databases (preferred if, for example, you use AWS RDS)

MySQL only, or external databases.

Perform a PCF Backup

To perform a manual backup of your PCF deployment, see [Backing Up Pivotal Cloud Foundry](#).

Restore From a PCF Backup

For information about manually restoring from a previous backup of your PCF deployment, see [Restoring Pivotal Cloud Foundry from Backup](#).

Backing Up Pivotal Cloud Foundry

Page last updated:

This topic describes the procedure for manually backing up each critical backend Elastic Runtime component. Pivotal recommends frequently backing up your installation settings before making any changes to your PCF deployment, such as configuration of any tiles in Ops Manager.

To back up a deployment, export installation settings, download the BOSH Deployment Manifest, temporarily stop the Cloud Controller, create and export backup files for each critical backend component, and restart the Cloud Controller. It is also important to record your Cloud Controller Database encryption credentials which you will need if you contact Pivotal Support for help restoring your installation.

To restore your backup, see the [Restoring Pivotal Cloud Foundry from Backup](#) topic.

Record the Cloud Controller Database Encryption Credentials

From the [Installation Dashboard](#), select **Pivotal Elastic Runtime > Credentials** and locate the Cloud Controller section. 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

Export Installation Settings

Pivotal recommends that you back up your installation settings by exporting frequently. This option is only available after you have deployed at least one time. Always export an installation before following the steps in the [Import Installation Settings](#) section of the [Restoring Pivotal Cloud Foundry from Backup](#) topic.

 **Note:** Exporting your installation only backs up your installation settings. 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**.

Export installation settings exports the current PCF installation settings and assets. When you export an installation, the exported file contains the base VM images, all necessary packages, and references to the installation IP addresses. As a result, an exported installation file can exceed 5 GB in size.

◀ Installation Dashboard

🔒 Settings

Decryption Passphrase

Authentication Method

External API Access

Proxy Settings

Export Installation Settings

Advanced

Export Installation

⚠️ Upgrading Ops Manager may block the upgrade path for certain products.
Please read [Upgrading Ops Manager](#) and verify an upgrade path exists for your installed products.

Export Installation Settings

Note: For versions of Ops Manager 1.3 or older, the process of archiving files for export may exceed the timeout limit of 600 seconds and result in a [500](#) error. To resolve this issue, you can manually increase the timeout value, or assign additional resources to the Ops Manager VM to improve performance. For more information, see the [Pivotal Support Knowledge Base](#).

Target the BOSH Director

1. Install Ruby and the [BOSH CLI Ruby gem](#) on a machine outside of your PCF deployment.
2. From the Installation Dashboard in Ops Manager, select **Ops Manager Director > Status** and record the IP address listed for the Director.

◀ Installation Dashboard

Ops Manager Director

Settings Status Credentials Logs

JOB	INDEX	IPS	CLOUD
Ops Manager Director	0	10.0.0.3	Vm 90 81 d

You access the BOSH Director using this IP address.

3. Click **Credentials** and record the Director credentials.

[Installation Dashboard](#)

Ops Manager Director

Settings Status **Credentials**

JOB	NAME	CREDENTIALS
Ops Manager Director	Vm Credentials	Link to Credential
	Agent Credentials	Link to Credential
	Registry Credentials	Link to Credential
	Director Credentials	Link to Credential

- From the command line, run `bosh target` to log into the BOSH Director using the IP address and credentials that you recorded:

```
$ bosh target 192.0.2.3
Target set to 'microbosh-1234abcd1234abcd1234'
Your username: director
Enter password: *****
Logged in as 'director'
```

 **Note:** If `bosh target` does not prompt you for your username and password, run `bosh login`.

Download BOSH Manifest

- Install Ruby and the [BOSH CLI Ruby gem](#) on a machine outside of your [Pivotal Cloud Foundry](#) (PCF) system deployment.
- 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.

[Installation Dashboard](#)

Ops Manager Director

Settings Status **Credentials Logs**

JOB	INDEX	IPS	CL
Ops Manager Director	0	10.0.0.3	vr 9 8 d

- Click **Credentials** and record the Director credentials.

[◀ Installation Dashboard](#)

Ops Manager Director

Settings Status **Credentials**

JOB	NAME	CREDENTIALS
Ops Manager Director	Vm Credentials	Link to Credential
	Agent Credentials	Link to Credential
	Registry Credentials	Link to Credential
	Director Credentials	Link to Credential

- From the command line, target the BOSH Director using the IP address and credentials that you recorded:

```
$ bosh target 10.0.0.3
Target set to 'microbosh-1234abcd1234abcd1234'
Your username: director
Enter password: *****
Logged in as 'director'
```

Note: If `bosh target` does not prompt you for your username and password, run `bosh login`.

- Run `bosh deployments` to identify the name of your current BOSH deployment:

```
$ bosh deployments
+-----+-----+
| Name | Release(s) | Stemcell(s) |
+-----+-----+
| cf-example | cf-mysql/10 | bosh-vsphere-esxi-ubuntu-trusty-go_agent/2690.3 |
|          | cf/183.2 |           |
+-----+-----+
```

- Run `bosh download manifest DEPLOYMENT-NAME LOCAL-SAVE-NAME` to download and save each BOSH deployment manifest. You need this manifest to locate information about your databases. For each manifest, you will need to repeat these instructions. Replace `DEPLOYMENT-NAME` with the name of the current BOSH deployment. For this procedure, use `cf.yml` as the `LOCAL-SAVE-NAME`.

```
$ bosh download manifest cf-example cf.yml
Deployment manifest saved to 'cf.yml'
```

Back Up Critical Backend Components

Your Elastic Runtime deployment contains several critical data stores that must be present for a complete restore. This section describes the procedure for backing up the databases and the servers associated with your PCF installation.

You must back up each of the following:

- Cloud Controller Database
- UAA Database
- WebDAV Server
- Pivotal MySQL Server

Note: If you are running PostgreSQL and are on the default internal databases, follow the instructions below. If you are running your databases or filestores externally, disregard instructions for backing up the Cloud Controller, and UAA Databases, and ensure that you back up your external databases and filestores.

Note: To follow the backup instructions below, your network must be configured to allow access to the BOSH Director VM from your

Your local machine. If you do not have local administrator access, use the `scp` command to copy the TAR file to the BOSH Director VM. For example: `scp vcap@192.0.2.10:webdav.tar.gz \ and vcap@192.0.2.3:/webdav.tar.gz`

Stop Cloud Controller

- From a command line, run `bosh deployment DEPLOYMENT-MANIFEST` to select your PCF deployment. The manifest is located in `/var/tempest/workspaces/default/deployments/` on the Ops Manager VM. For example:

```
$ bosh deployment /var/tempest/workspaces/default/deployments/cf-bd784.yml
Deployment set to '/var/tempest/workspaces/default/deployments/cf-bd784.yml'
```

- Run `bosh vms CF-DEPLOYMENT-NAME` to view a list of VMs in your PCF deployment. `CF-DEPLOYMENT-NAME` corresponds to the name of your PCF release deployment, which is also the filename of your manifest file without the `.yml` ending. For example:

```
$ bosh vms cf-bd784
+-----+-----+-----+
| Job/index | State | Resource Pool | IPs |
+-----+-----+-----+
| ccdp-partition-bd784/0 | running | ccdp-partition-bd784 | 10.85.xx.xx |
| cloud_controller-partition-bd784/0 | running | cloud_controller-partition-bd784 | 10.85.xx.xx |
| cloud_controller_worker-partition-bd784/0 | running | cloud_controller-worker-partition-bd784 | 10.85.xx.xx |
| clock_global-partition-bd784/0 | running | clock_global-partition-bd784 | 10.85.xx.xx |
| nats-partition-bd784/0 | running | nats-partition-bd784 | 10.85.xx.xx |
| router-partition-bd784/0 | running | router-partition-bd784 | 10.85.xx.xx |
| uaa-partition-bd784/0 | running | uaa-partition-bd784 | 10.85.xx.xx |
+-----+-----+-----+
```

- Run `bosh stop SELECTED-VM` for each Cloud Controller VM. For example, stop the `cloud_controller-partition-bd784` and `cloud_controller_worker-partition-bd784` VMs.

```
$ bosh stop cloud_controller-partition-bd784
Acting as user 'director'...
You are about to stop cloud_controller-partition-bd784/0
Detecting deployment changes
-----
Stop cloud_controller-partition-bd784/0? (type 'yes' to continue)

$ bosh stop cloud_controller_worker-partition-bd784
Acting as user 'director'...
You are about to stop cloud_controller_worker-partition-bd784/0
Detecting deployment changes
-----
Stop cloud_controller_worker-partition-bd784/0? (type 'yes' to continue)
```

Do not stop the `ccdb-partition` VM, which is the backend database for Cloud Controller if you opted to use PostgreSQL in your database deployment.

Back Up the Cloud Controller Database

You can follow these instructions only if you are using a PostgreSQL database.

- Use `bosh ssh` to SSH into the Cloud Controller Database VM:
 - Run `ssh-keygen -t rsa` to provide BOSH with the correct public key.
 - Accept the defaults.
 - Run `bosh ssh`.
 - When prompted to choose an instance, enter the number that corresponds to the VM beginning with `ccdb`.
 - Create a password for the temporary user that the `bosh ssh` command creates. Use this password if you need `sudo` access in this session.
- Run `pg_dump` to export the database:

```
$ pg_dump -U admin -p 2544 ccdb > /tmp/ccdb.sql
```
- Run `bosh scp` to copy the exported database to your BOSH client.

```
$ bosh scp --download ccdb/0 /tmp/ccdb.sql .
```

4. Exit from the Cloud Controller database VM.

Back Up the UAA Database

 **Note:** Follow these instructions only if you are using a PostgreSQL database.

1. Use `bosh ssh` to SSH into the UAA Database VM:
 - a. Run `ssh-keygen -t rsa` to provide BOSH with the correct public key.
 - b. Accept the defaults.
 - c. Run `bosh ssh`.
 - d. When prompted to choose an instance, enter the number that corresponds to the VM beginning with `uaadb`.
 - e. Create a password for the temporary user that the `bosh ssh` command creates. Use this password if you need `sudo` access in this session.
2. Run `pg_dump` to export the database:

```
$ pg_dump -U admin -p 2544 uaa > /tmp/uaa.sql
```
3. Run `bosh scp` to copy the exported database to your BOSH client:

```
$ bosh scp --download uaadb/0 /tmp/uaa.sql .
```
4. Exit from the UAA database VM.

Back Up WebDAV Server

1. In the BOSH deployment manifest, locate the `nfs_server` component and record the address:

```
nfs_server:  
address: 192.0.2.10  
network: 192.0.2.0/24  
syslog_aggregator:  
address:  
port:
```

 **Note:** The job name associated with the WebDAV server is `nfs_server` for historical reasons. The server is not based on NFS.

2. From the **Installation Dashboard** in Ops Manager, select **Elastic Runtime** and click **Credentials > Link to Credential**. Record the File Storage server VM credentials.

File Storage	VM Credentials	Link to Credential
--------------	----------------	--------------------

3. SSH into the WebDAV server VM and create a TAR file:

```
$ ssh vcap@192.0.2.10 'cd /var/vcap/store && tar cz shared' > webdav.tar.gz
```

 **Note:** The TAR file that you create to back up WebDAV server might be large. To estimate the size of the TAR file before you create it, run the following command: `ssh vcap@192.0.2.10 tar -cf - /dir/to/archive/ | wc -c`

Back Up Pivotal MySQL Server

 **Note:** The Elastic Runtime deploy contains an embedded MySQL Server that serves as the data store for the Application Usage Events, Notifications, and Autoscaler services. If you are using an internal MySQL, this will also include the Cloud Controller and UAA.

There are two ways to backup the MySQL Server:

- **Manual backup:** If you have not set up automatic backups, you need to do a manual backup of your MySQL server.
- Automatic backup: If you set automatic backup in your ERT configuration, you do not need to manually backup your MySQL Server. Automatic backup requires S3-compatible blobstores. For more information, see:
 - AWS: [Configure Internal MySQL](#)
 - OpenStack: [Configure Internal MySQL](#)
 - vSphere [Configure Internal MySQL](#)

Backing up MySQL Server Manually

1. From the Installation Dashboard in Ops Manager, select **Pivotal Elastic Runtime**. Click **Status** and record the IP address of the first instance of **MySQL Server**.

JOB	INDEX	IPS	AZ	CID
Consul	0	10.85.10.96	first-az	vm-e390448b-1f94-44f5-8a26-4675219e064b
NATS	0	10.85.10.97	first-az	vm-6490210f-ecea-4f83-8b8f-8529b132a6a7
etcd	0	10.85.10.98	first-az	vm-228f110c-5b85-4f28-a5af-23a32ed12812
Diego BBS	0	10.85.10.92	first-az	vm-e84a441d-924b-4b97-ae76-e6fb95578bde
File Storage	0	10.85.10.100	first-az	vm-e7a7ee0b-e49a-4c42-bd94-68b58ee7e3c1
MySQL Proxy	0	10.85.10.101	first-az	vm-6230f0b2-e20b-4c8c-856d-b0c11782970c
MySQL Server	0	10.85.10.102	first-az	vm-fe60305f-4ddb-4251-8c74-2b15b655ffc2

2. Click **Credentials** and record the Mysql Admin Credentials of **MySQL Server**.

MySQL Server	VM Credentials	Link to Credential
	Mysql Admin Credentials	Link to Credential

3. SSH into the MySQL database VM as the admin using the IP address and password recorded in the previous steps.

4. Run the following command to dump the data from the p-mysql database and save it:

```
$ mysqldump -u root -p -h 10.0.1.26 --all-databases > user_databases.sql
```

5. Run `scp` to copy the exported database to your local machine.

```
$ scp vcap@10.0.1.26:~/user_databases.sql .
```

Start Cloud Controller

1. Run `bosh vms` to view a list of VMs in your selected deployment. The names of the Cloud Controller VMs begin with `cloud_controller`.

```
$ bosh vms
+-----+-----+-----+
| Job/index | State | Resource Pool | IPs | 
+-----+-----+-----+
| cloud_controller-partition-bd784/0 | failing | cloud_controller-partition-bd784 | 10.85.xx.xx |
| cloud_controller_worker-partition-bd784/0 | running | cloud_controller-partition-bd784 | 10.85.xx.xx |
| clock_global-partition-bd784/0 | running | clock_global-partition-bd784 | 10.85.xx.xx |
| nats-partition-bd784/0 | running | nats-partition-bd784 | 10.85.xx.xx |
| router-partition-bd784/0 | running | router-partition-bd784 | 10.85.xx.xx |
| uaa-partition-bd784/0 | running | uaa-partition-bd784 | 10.85.xx.xx |
+-----+-----+-----+
```

2. BOSH indicates a stopped VM with the state: `failing`. Run `bosh start SELECTED-VM` for each stopped Cloud Controller VM.

```
$ bosh start cloud_controller-partition-bd784
Processing deployment manifest
-----
Processing deployment manifest
-----
You are about to start cloud_controller-partition-bd784/0
Processing deployment manifest
-----
Detecting deployment changes
-----
Start cloud_controller-partition-bd784/0? (type 'yes' to continue): yes
Performing 'start cloud_controller-partition-bd784/0...
Director task 1428
Started preparing deployment
Started preparing deployment > Binding deployment. Done (00:00:00)
Started preparing deployment > Binding releases. Done (00:00:00)
Started preparing deployment > Binding existing deployment. Done (00:00:01)
Started preparing deployment > Binding resource pools. Done (00:00:00)
Started preparing deployment > Binding stemcells. Done (00:00:00)
Started preparing deployment > Binding templates. Done (00:00:00)
Started preparing deployment > Binding properties. Done (00:00:00)
Started preparing deployment > Binding unallocated VMs. Done (00:00:00)
Started preparing deployment > Binding instance networks. Done (00:00:00)
Done preparing deployment (00:00:01)
Started preparing package compilation > Finding packages to compile. Done (00:00:01)
Started preparing dns > Binding DNS. Done (00:00:00)
Started preparing configuration > Binding configuration. Done (00:00:13)
Started updating job cloud_controller-partition-bd784/0 > cloud_controller-partition-bd784/0 (canary)^@. Done (00:01:44)
Task 1428 done
Started 2015-02-25 17:54:28 UTC
Finished 2015-02-25 17:56:27 UTC
Duration 00:01:59
cloud_controller-partition-bd784/0 has been started
```

Follow the steps in the [Restoring Pivotal Cloud Foundry from Backup](#) topic to restore a backup, import an installation to restore your settings, or to share your settings with another user.

Restoring Pivotal Cloud Foundry from Backup

Page last updated:

This topic describes the procedure for manually restoring your Pivotal Cloud Foundry (PCF) deployment from a backup. To create a backup, see the [Backing Up Pivotal Cloud Foundry](#) topic.

Note: You can also use the CF Ops automation utility to perform a restore of your PCF backups. See the [CF Ops User Guide](#) for more information.

To restore a deployment, you must import installation settings, temporarily stop the Cloud Controller, restore the state of each critical backend component from its backup file, and restart the Cloud Controller.

To perform these steps, you need the BOSH manifest to locate your critical backend components. BOSH manifests are automatically downloaded to the Ops Manager virtual machine. However, if you are using a separate jumpbox, you must manually download the BOSH deployment manifest.

Note: The procedure described in this topic restores a running PCF deployment to the state captured by backup files. This procedure does not deploy PCF. See the [Installing PCF Guide](#) for information.

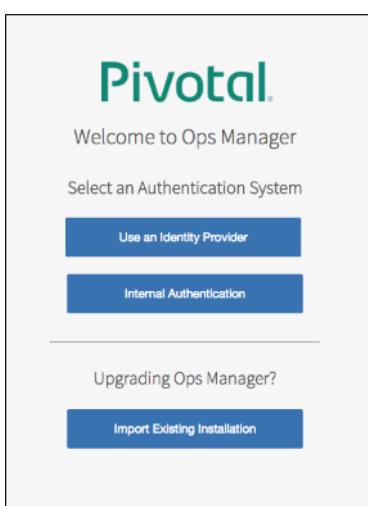
Import Installation Settings

Note: Pivotal recommends that you export your installation settings before importing from a backup. See the [Export Installation Settings](#) section of the [Backing Up Pivotal Cloud Foundry](#) topic for more information.

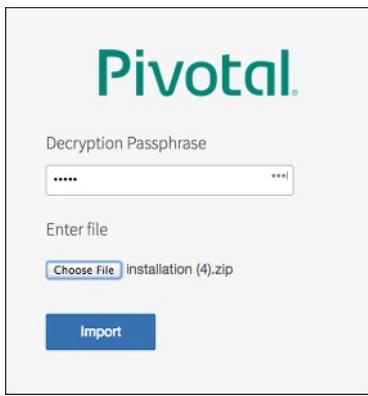
Import installation settings imports the settings and assets of an existing PCF installation. Importing an installation overwrites any existing installation. You must provision a new Ops Manager to import settings.

Follow the steps below to import installation settings.

1. Select and follow the instruction below for your IaaS to deploy the new Ops Manager VM.
 - [Launching an Ops Manager Director Instance on AWS](#)
 - [Launching an Ops Manager Director Instance on Azure](#)
 - [Launching an Ops Manager Director Instance on GCP](#)
 - [Provisioning the OpenStack Infrastructure](#)
 - [Deploying Operations Manager to vSphere](#)
2. When redirected to the [Welcome to Ops Manager](#) page, select **Import Existing Installation**.



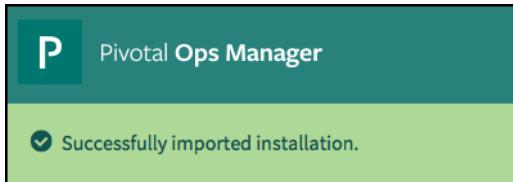
3. In the import panel, perform the following tasks:
 - Enter your **Decryption Passphrase**.
 - Click **Choose File** and browse to the installation zip file that you exported in the [Export Installation Settings](#) section of this topic.



4. Click **Import**.

 **Note:** Some browsers do not provide feedback on the status of the import process, and may appear to hang.

5. A “Successfully imported installation” message appears upon completion.



6. Click **Apply Changes**. This immediately imports and applies upgrades to all tiles in a single transaction.

Restore BOSH Using Ops Manager

Follow the steps below to restore BOSH using Ops Manager.

1. From the **Product Installation Dashboard**, click the **Ops Manager Director** tile.
2. Make a change to your configuration to trigger a new deployment. For example, you can adjust the number of NTP servers in your deployment. Choose a change in configuration that suits your specific deployment.
3. Follow the instructions in the [SSH into Ops Manager](#) topic. The following example assumes an Amazon Web Services deployment:

```
$ ssh -i ops_mgr.pem ubuntu@OPS-MGR-IP
```

4. Delete the `bosh-deployments.yml` file. Deleting the `bosh-deployments.yml` file causes Ops Manager to treat the deploy as a new deployment, recreating missing Virtual Machines (VMs), including BOSH. The new deployment ignores existing VMs such as your Pivotal Cloud Foundry deployment.

```
$ sudo rm /var/tempest/workspaces/default/deployments/bosh-deployments.yml
```

5. Rename, move, or delete the `bosh-state.json` file. Removing this file causes Ops Manager to treat the deploy as a new deployment, recreating missing VMs, including BOSH. The new deployment ignores existing VMs such as your Pivotal Cloud Foundry deployment.

```
$ cd /var/tempest/workspaces/default/deployments/  
$ sudo mv bosh-state.json bosh-state.old
```

6. Return to the **Product Installation Dashboard** and click **Apply Changes**.

Target the BOSH Director

1. Install Ruby and the [BOSH CLI Ruby gem](#) on a machine outside of your PCF deployment.

2. From the Installation Dashboard in Ops Manager, select **Ops Manager Director > Status** and record the IP address listed for the Director.

JOB	INDEX	IPS	CLOUD
Ops Manager Director	0	10.0.0.3	Vm 90 83 d

You access the BOSH Director using this IP address.

3. Click **Credentials** and record the Director credentials.

JOB	NAME	CREDENTIALS
Ops Manager Director	Vm Credentials	Link to Credential
	Agent Credentials	Link to Credential
	Registry Credentials	Link to Credential
	Director Credentials	Link to Credential

4. From the command line, run `bosh target` to log into the BOSH Director using the IP address and credentials that you recorded:

```
$ bosh target 192.0.2.3
Target set to 'microbosh-1234abcd1234abcd1234'
Your username: director
Enter password: *****
Logged in as 'director'
```

Note: If `bosh target` does not prompt you for your username and password, run `bosh login`.

Download BOSH Manifest

1. Install Ruby and the [BOSH CLI Ruby gem](#) on a machine outside of your [Pivotal Cloud Foundry](#) (PCF) system deployment.
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.

JOB	INDEX	IPS	CLOUD
Ops Manager Director	0	10.0.0.3	Vm 90 83 d

3. Click **Credentials** and record the Director credentials.

JOB	NAME	CREDENTIALS
Ops Manager Director	Vm Credentials	Link to Credential
	Agent Credentials	Link to Credential
	Registry Credentials	Link to Credential
	Director Credentials	Link to Credential

4. From the command line, target the BOSH Director using the IP address and credentials that you recorded:

```
$ bosh target 10.0.0.3
Target set to `microbosh-1234abcd1234abcd1234'
Your username: director
Enter password: *****
Logged in as `director'
```

Note: If `bosh target` does not prompt you for your username and password, run `bosh login`.

5. Run `bosh deployments` to identify the name of your current BOSH deployment:

```
$ bosh deployments
+-----+-----+
| Name | Release(s) | Stemcell(s) |
+-----+-----+
| cf-example | cf-mysql/10 | bosh-vsphere-esxi-ubuntu-trusty-go_agent/2690.3 |
|          | cf/183.2   |           |
+-----+-----+
```

6. Run `bosh download manifest DEPLOYMENT-NAME LOCAL-SAVE-NAME` to download and save each BOSH deployment manifest. You need this manifest to locate information about your databases. For each manifest, you will need to repeat these instructions. Replace `DEPLOYMENT-NAME` with the name of the current BOSH deployment. For this procedure, use `cf.yml` as the `LOCAL-SAVE-NAME`.

```
$ bosh download manifest cf-example cf.yml
Deployment manifest saved to `cf.yml'
```

Restore Critical Backend Components

Your Elastic Runtime deployment contains several critical data stores that must be present for a complete restore. This section describes the procedure for restoring the databases and servers associated with your PCF installation.

You must restore each of the following:

- Cloud Controller Database
- UAA Database
- WebDAV Server
- Pivotal MySQL Server

Note: If you are running PostgreSQL and are on the default internal databases, follow the instructions below. If you are running your databases or filestores externally, disregard instructions for restoring the Cloud Controller and UAA Databases.

Stop Cloud Controller

- From a command line, run `bosh deployment DEPLOYMENT-MANIFEST` to select your PCF deployment. The manifest is located in `/var/tempest/workspaces/default/deployments/` on the Ops Manager VM. For example:

```
$ bosh deployment /var/tempest/workspaces/default/deployments/cf-bd784.yml
Deployment set to '/var/tempest/workspaces/default/deployments/cf-bd784.yml'
```

- Run `bosh vms CF-DEPLOYMENT-NAME` to view a list of VMs in your PCF deployment. `CF-DEPLOYMENT-NAME` corresponds to the name of your PCF release deployment, which is also the filename of your manifest file without the `.yml` ending. For example:

Job/index	State	Resource Pool	IPs
ccdb-partition-bd784/0	running	ccdb-partition-bd784	10.85.xx.xx
cloud_controller-partition-bd784/0	running	cloud_controller-partition-bd784	10.85.xx.xx
cloud_controller_worker-partition-bd784/0	running	cloud_controller-partition-bd784	10.85.xx.xx
clock_global-partition-bd784/0	running	clock_global-partition-bd784	10.85.xx.xx
nats-partition-bd784/0	running	nats-partition-bd784	10.85.xx.xx
router-partition-bd784/0	running	router-partition-bd784	10.85.xx.xx
uaa-partition-bd784/0	running	uaa-partition-bd784	10.85.xx.xx

- Run `bosh stop SELECTED-VM` for each Cloud Controller VM. For example, stop the `cloud_controller-partition-bd784` and `cloud_controller_worker-partition-bd784` VMs.

```
$ bosh stop cloud_controller-partition-bd784
Acting as user 'director'...
You are about to stop cloud_controller-partition-bd784/0
Detecting deployment changes
-----
Stop cloud_controller-partition-bd784/0? (type 'yes' to continue)

$ bosh stop cloud_controller_worker-partition-bd784
Acting as user 'director'...
You are about to stop cloud_controller_worker-partition-bd784/0
Detecting deployment changes
-----
Stop cloud_controller_worker-partition-bd784/0? (type 'yes' to continue)
```

Do not stop the `ccdb-partition` VM, which is the backend database for Cloud Controller if you opted to use PostgreSQL in your database deployment.

Restore the Cloud Controller Database

 **Note:** Follow these instructions only if you are using a PostgreSQL database.

Use the Cloud Controller Database (CCDB) password and IP address to restore the Cloud Controller Database by following the steps detailed below. Find the IP address in your BOSH deployment manifest. To find your password in the Ops Manager **Installation Dashboard**, select **Elastic Runtime** and click **Credentials>Link to Credential**.

- Use `scp` to send the Cloud Controller Database backup file to the Cloud Controller Database VM.

```
$ scp ccdb.sql vcap@YOUR-CCDB-VM-IP-ADDRESS:~/
```

- SSH into the Cloud Controller Database VM.

```
$ ssh vcap@YOUR-CCDB-VM-IP
```

- Log in to the `psql` client

```
$ /var/vcap/data/packages/postgres/5.1/bin/psql -U vcap -p 2544 ccdb
```

- Drop the database schema and create a new one to replace it.

```
ccdb=# drop schema public cascade;
ccdb=# create schema public;
```

5. Restore the database from the backup file.

```
$ /var/vcap/data/packages/postgres/5.1/bin/psql -U vcap -p 2544 ccdb < ~/ccdb.sql
```

Restore UAA Database

 **Note:** Follow these instructions only if you are using a PostgreSQL database.

Drop the UAA Database tables

1. Find your UAA Database VM ID. To view all VM IDs, run `bosh vms` from a command line:

```
$ bosh vms
```

2. SSH into the UAA Database VM using the vcap user and password. If you do not have this information recorded, find it in the Ops Manager **Installation Dashboard**. Click the **Elastic Runtime** tile and select **Credentials>Link to Credential**.

```
$ ssh vcap@YOUR-UAADB-VM-IP-ADDRESS
```

3. Run `find /var/vcap | grep 'bin/psql'` to find the locally installed psql client on the UAA Database VM.

```
$ vcap@198.51.100.101:~# find /var/vcap | grep 'bin/psql'
/var/vcap/data/packages/postgres/5.1/bin/psql
```

4. Log in to the psql client:

```
$ vcap@198.51.100.101:~# /var/vcap/data/packages/postgres/5.1/bin/psql -U vcap -p 2544 uaa
```

5. Run the following commands to drop the tables:

```
drop schema public cascade;
create schema public;
\q
```

6. Exit the UAA Database VM.

```
$ exit
```

Restore the UAA Database from its backup state

 **Note:** Follow these instructions only if you are using a PostgreSQL database.

1. Use the UAA Database password and IP address to restore the UAA Database by running the following commands. You can find the IP address in your BOSH deployment manifest. To find your password in the Ops Manager **Installation Dashboard**, select **Elastic Runtime** and click **Credentials>Link to Credential**.
2. Use `scp` to copy the database backup file to the UAA Database VM.

```
$ scp uaa.sql vcap@YOUR-UAADB-VM-IP-ADDRESS:~/
```

3. SSH into the UAA Database VM.

```
$ ssh vcap@YOUR-UAADB-VM-IP-ADDRESS
```

4. Restore the database from the backup file.

```
$ /var/vcap/data/packages/postgres/5.1/bin/psql -U vcap -p 2544 uaa < ~/uaa.sql
```

Restore WebDAV

Use the File Storage password and IP address to restore the WebDAV server by following the steps detailed below. Find the IP address in your BOSH deployment manifest. To find your password in the Ops Manager **Installation Dashboard**, select **Elastic Runtime** and click **Credentials>Link to Credential**.

1. Run `ssh` YOUR-WEBDAV-VM-IP-ADDRESS to enter the WebDAV VM.

```
$ ssh vcap@192.0.2.10
```

2. Log in as root user. When prompted for a password, enter the vcap password you used to `ssh` into the VM:

```
$ sudo su
```

3. Temporarily change the permissions on `/var/vcap/store` to add write permissions for all.

```
$ chmod a+w /var/vcap/store
```

4. Use `scp` to send the WebDAV backup tarball to the WebDAV VM from your local machine.

```
$ scp webdav.tar.gz vcap@YOUR-WEBDAV-VM-IP-ADDRESS:/var/vcap/store
```

5. Navigate into the `store` folder on the WebDAV VM.

```
$ cd /var/vcap/store
```

6. Decompress and extract the contents of the backup archive.

```
$ tar zxf webdav.tar.gz
```

7. Change the permissions on `/var/vcap/store` to their prior setting.

```
$ chmod a-w /var/vcap/store
```

8. Exit the WebDAV VM.

```
$ exit
```

Restore MySQL Database

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

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

1. From the same VM you used to perform the manual backup, or a similar VM, restore from the data dump:

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

2. Use `scp` to send the MySQL Database backup file to the MySQL Database VM. Find the IP address in your BOSH deployment manifest. To find your password in the Ops Manager Installation Dashboard, select **Elastic Runtime** and click **Credentials**.

```
$ scp user_databases.sql vcap@YOUR-MYSQL-VM-IP-ADDRESS:~/
```

3. SSH into the MySQL Database VM.

```
$ ssh vcap@YOUR-MYSQL-VM-IP
```

4. Use the MySQL password and IP address to enable the creation of tables using any storage engine.

```
$ mysql -h YOUR-MYSQL-SERVER-IP -u root -p -e "SET GLOBAL enforce_storage_engine=NULL;"
```

5. Use the MySQL password and IP address to restore the MySQL database by running the following command.

```
$ mysql -h YOUR-MYSQL-SERVER-IP -u root -p < ~/.user_databases.sql
```

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

```
$ mysql -h YOUR-MYSQL-SERVER-IP -u root -p -e "SET GLOBAL enforce_storage_engine='InnoDB';"
```

7. Log in to the MySQL client and flush privileges.

```
$ mysql -u root -p -h
mysql> flush privileges;
```

Start Cloud Controller

1. Run `bosh vms` to view a list of VMs in your selected deployment. The names of the Cloud Controller VMs begin with `cloud_controller`.

```
$ bosh vms
+-----+-----+-----+
| Job/index | State | Resource Pool | IPs |
+-----+-----+-----+
| cloud_controller-partition-bd784/0 | failing | cloud_controller-partition-bd784 | 10.85.xx.xx |
| cloud_controller_worker-partition-bd784/0 | running | cloud_controller-partition-bd784 | 10.85.xx.xx |
| clock_global-partition-bd784/0 | running | clock_global-partition-bd784 | 10.85.xx.xx |
| nats-partition-bd784/0 | running | nats-partition-bd784 | 10.85.xx.xx |
| router-partition-bd784/0 | running | router-partition-bd784 | 10.85.xx.xx |
| uaa-partition-bd784/0 | running | uaa-partition-bd784 | 10.85.xx.xx |
+-----+-----+-----+
```

2. BOSH indicates a stopped VM with the state: `failing`. Run `bosh start SELECTED-VM` for each stopped Cloud Controller VM.

```
$ bosh start cloud_controller-partition-bd784
Processing deployment manifest
-----
Processing deployment manifest
-----
You are about to start cloud_controller-partition-bd784/0
Processing deployment manifest
-----
Detecting deployment changes
-----
Start cloud_controller-partition-bd784/0? (type 'yes' to continue): yes
Performing `start cloud_controller-partition-bd784/0'...
Director task 1428
Started preparing deployment
Started preparing deployment > Binding deployment. Done (00:00:00)
Started preparing deployment > Binding releases. Done (00:00:00)
Started preparing deployment > Binding existing deployment. Done (00:00:01)
Started preparing deployment > Binding resource pools. Done (00:00:00)
Started preparing deployment > Binding stemcells. Done (00:00:00)
Started preparing deployment > Binding templates. Done (00:00:00)
Started preparing deployment > Binding properties. Done (00:00:00)
Started preparing deployment > Binding unallocated VMs. Done (00:00:00)
Started preparing deployment > Binding instance networks. Done (00:00:00)
Done preparing deployment (00:00:01)
Started preparing package compilation > Finding packages to compile. Done (00:00:01)
Started preparing dns > Binding DNS. Done (00:00:00)
Started preparing configuration > Binding configuration. Done (00:00:13)
Started updating job cloud_controller-partition-bd784/0 > cloud_controller-partition-bd784/0 (canary)^@. Done (00:01:44)
Task 1428 done
Started 2015-02-25 17:54:28 UTC
Finished 2015-02-25 17:56:27 UTC
Duration 00:01:59
cloud_controller-partition-bd784/0 has been started
```

Monitoring Virtual Machines in Pivotal Cloud Foundry

Page last updated:

 **Note:** Pivotal Cloud Foundry (PCF) for vCloud Air and vCloud Director is deprecated and availability is restricted to existing customers. Contact [Support](#) for more information.

This topic covers strategies for monitoring virtual machine (VM) status and performance in [Pivotal Cloud Foundry](#) (PCF).

Monitoring VMs Using the Ops Manager Interface

Click any product tile and select the **Status** tab to view monitoring information.

Pivotal Elastic Runtime															
Settings	Status	Credentials	Logs												
JOB	INDEX	IPS	CID	LOAD AVG15	CPU	MEMORY	SWAP	SYSTEM DISK	EPHEM. DISK	PERS. DISK	LOGS				
HAProxy	0	10.0.0.254	vm-9985a13c-106a-48d1-a3de-d0e0e816c857	0.06%	0.1%	9.6%	0.0%	41%	5%	N/A					
NATS	0	10.0.0.5	vm-dee49615-aea8-4f4f-bf0f-b1060083ddef	0.12%	0.1%	9.7%	0.0%	41%	21%	N/A					
			vm-6d43e59e-												

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.

Pivotal Cloud Foundry Troubleshooting Guide

Page last updated:

 **Note:** Pivotal Cloud Foundry (PCF) for vCloud Air and vCloud Director is deprecated and availability is restricted to existing customers. Contact [Support](#) for more information.

This guide provides help with diagnosing and resolving issues encountered during a [Pivotal Cloud Foundry](#) (PCF) installation. For help troubleshooting issues that are specific to PCF deployments on VMware vSphere, refer to the topic on [Troubleshooting Ops Manager for VMware vSphere](#).

An install or update can fail for many reasons. Fortunately, the system tends to heal or work around hardware or network faults. By the time you click the [Install](#) or [Apply Changes](#) button again, the problem may be resolved.

Some failures produce only generic errors like `Exited with 1`. In cases like this, where a failure is not accompanied by useful information, retry clicking [Install](#) or [Apply Changes](#).

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.

Besides whether products install successfully or not, an important area to consider when troubleshooting 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 Elastic Runtime 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 in troubleshooting. 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 Elastic Runtime 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 Elastic Runtime Components).

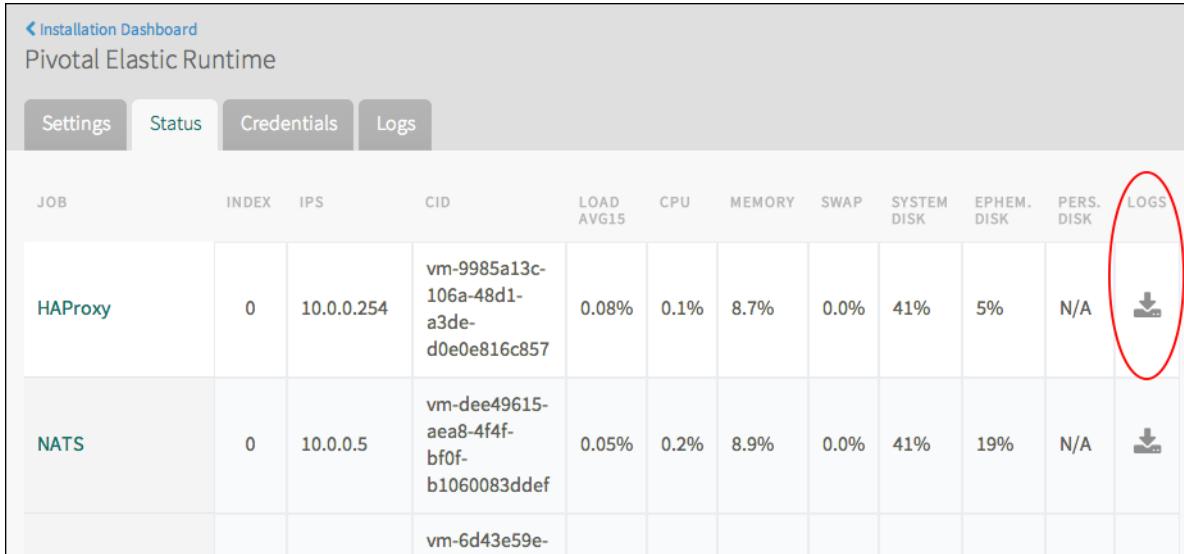
- Start with the largest and most recently updated files in the job log
- Identify logs that contain 'err' in the name
- Scan the file contents for a "failed" or "error" string

Viewing Logs for Elastic Runtime Components

To troubleshoot specific Elastic Runtime components by viewing their log files, browse to the Ops Manager interface and follow the procedure

below.

1. In Ops Manager, browse to the **Pivotal Elastic Runtime > Status** tab. In the **Job** column, locate the component of interest.
2. In the **Logs** column for the component, click the download icon.



JOB	INDEX	IPS	CID	LOAD AVG15	CPU	MEMORY	SWAP	SYSTEM DISK	EPHEM. DISK	PERS. DISK	LOGS
HAProxy	0	10.0.0.254	vm-9985a13c-106a-48d1-a3de-d0e0e816c857	0.08%	0.1%	8.7%	0.0%	41%	5%	N/A	
NATS	0	10.0.0.5	vm-dee49615-aea8-4f4f-bf0f-b1060083ddef	0.05%	0.2%	8.9%	0.0%	41%	19%	N/A	

3. Browse to the **Pivotal Elastic Runtime > Logs** tab.



FILENAME	UPDATED AT
Downloaded:	
/tmp/jobs_logs/diego_cell-partition-ee0b66b1415c8591855d-0-150e690af6dd.zip	2016-04-12 19:06:47 UTC
/tmp/jobs_logs/diego_database-partition-ee0b66b1415c8591855d-0-ab7bd3378ef2.zip	2016-04-12 19:09:30 UTC
Pending:	
/tmp/jobs_logs/diego_brain-partition-ee0b66b1415c8591855d-0-196edf23631b.zip	2016-04-12 19:09:34 UTC

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 Elastic Runtime 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 [Pivotal Elastic Runtime > 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.

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 **Elastic Runtime** 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-6aa9-4c39-8019-7024cfddde4",
"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.micrzbosh", 
"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:

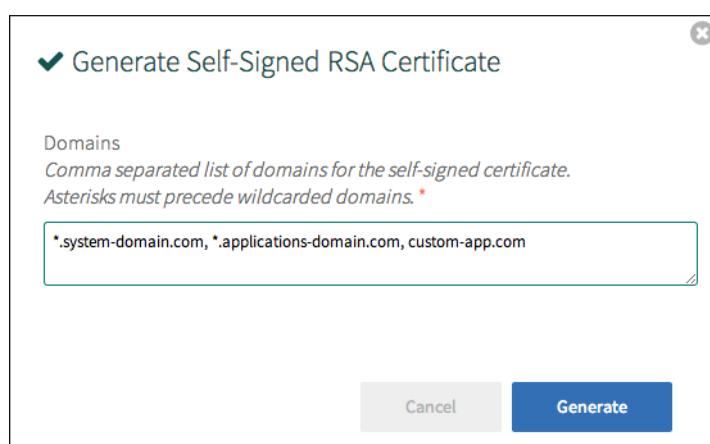
```
{"type": "step_finished", "id": "apps-manager.deploy"}  
/home/tempest-web/tempest/web/vendor/bundle/ruby/1.9.1/gems/mechanize-2.7.2/lib/mechanize/http/agent.rb:306:in  
`fetch': 403 => Net::HTTPForbidden for https://login.api.example.net/oauth/authorizeresponse_type=code&client_id=portal&redirect_uri=https%3...  
-- 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 **Elastic Runtime** 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 **Elastic Runtime > 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 Elastic Runtime for that product installation. If you attempt to install a product tile with an Elastic Runtime dependency before installing Elastic Runtime, 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 Elastic Runtime 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 Elastic Runtime Fails

If the DNS information for the PCF VM becomes incorrect after Ops Manager Director has been installed, installing Elastic Runtime 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 Elastic Runtime 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.

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 the [Pivotal Cloud Foundry Troubleshooting Guide](#).

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.

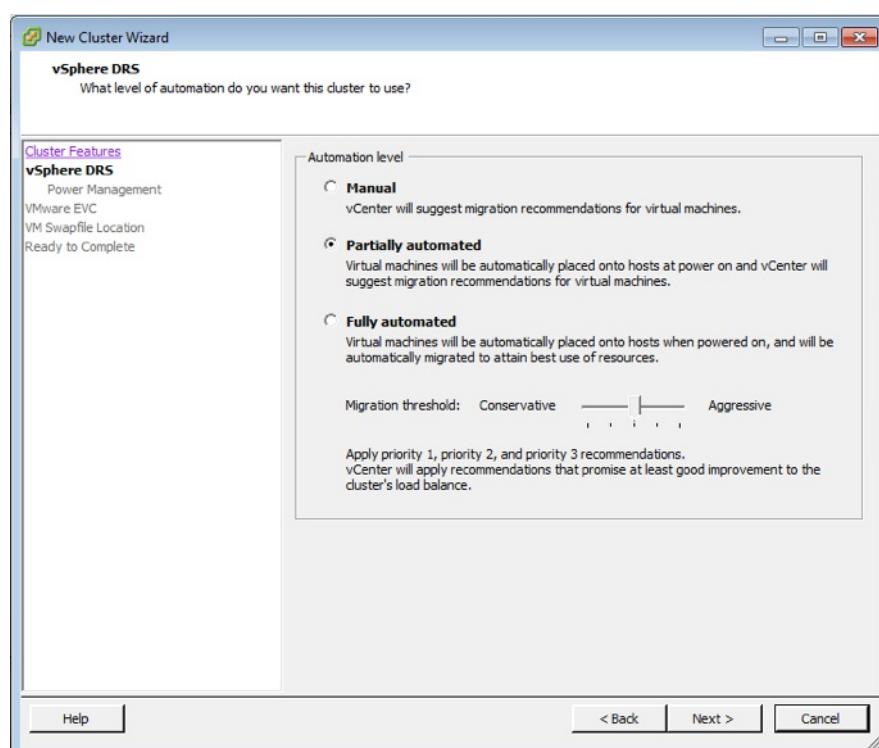
To resolve this issue, launch vCenter, navigate to **Administration > vCenter Server Settings > Statistics**, and reset the vCenter Statistics Interval Duration setting to 5 minutes.

BOSH Automated Installation Fails

Before starting an Elastic Runtime 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.



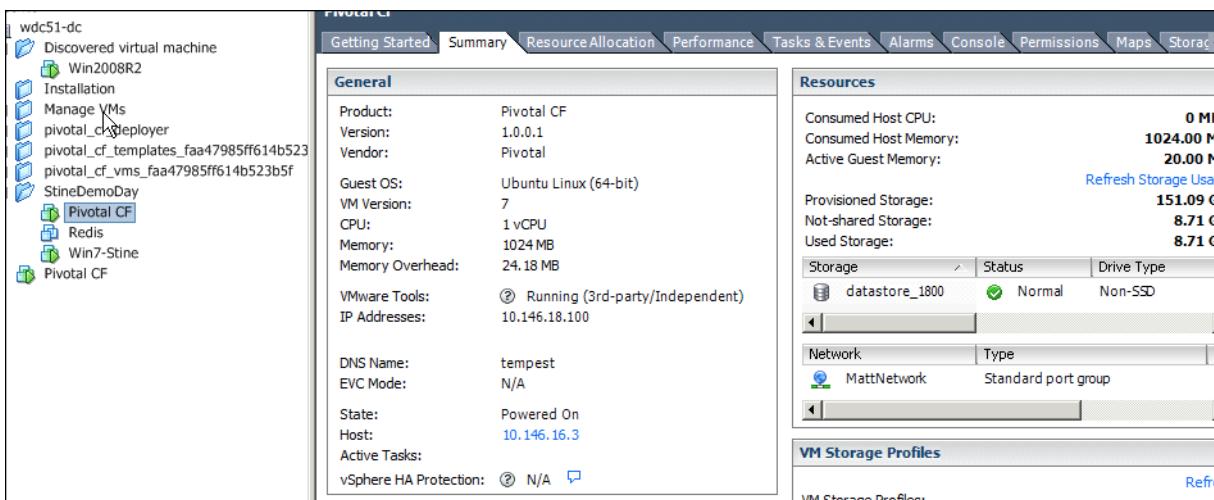
Ops Manager Loses Its IP Address After HA or Reboot

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 range during installation. BOSH uses these IPs during installation to compile and deploy VMs, install Elastic Runtime, and connect to services. We recommend that you allocate at least 36 dynamic IP addresses when deploying Ops Manager and Elastic Runtime.

Recovering MySQL from Elastic Runtime Downtime

Page last updated:

This topic describes the procedure for recovering a terminated Elastic Runtime 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`:

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

- All responsive nodes report the value of `wsrep_cluster_status` as `non-Primary`:

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

- All responsive nodes respond with `ERROR 1047` when 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

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

- Log into the BOSH director by running `bosh target DIRECTOR-URL` followed by `bosh login USERNAME PASSWORD`.
- Run `bosh deployments`.

```
$ bosh deployments
Acting as user 'director' on 'p-bosh-30c19bdd43c55c627d70'

+-----+-----+-----+-----+
| Name | Release(s) | Stemcell(s) | Cloud Config |
+-----+-----+-----+-----+
| cf-e82cbf44613594d8a155 | cf-autoscaling/28 | bosh-aws-xen-hvm-ubuntu-trusty-go_agent/3140 | none |
|   | cf-mysql/23 | | |
|   | cf/225 | | |
|   | diego/0.1441.0 | | |
|   | etcd/18 | | |
|   | garden-linux/0.327.0 | | |
|   | notifications-ui/10 | | |
|   | notifications/19 | | |
|   | push-apps-manager-release/397 | | |
+-----+-----+-----+-----+
```

3. Download the manifest.

```
$ bosh download manifest cf-e82cbf44613594d8a155 /tmp/cf.yml
Acting as user 'director' on deployment 'cf-e82cbf44613594d8a155' on 'p-bosh-30c19bdd43c55c627d70'
Deployment manifest saved to '/tmp/cf.yml'
```

4. Set BOSH to use the deployment manifest you downloaded.

```
$ bosh deployment /tmp/cf.yml
```

Step 2: Run the Bootstrap Errand

Elastic Runtime 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 instances` and examine the output.

- If the output of `bosh instances` shows the state of the jobs as `failing`, proceed to Scenario 1.

```
$ bosh instances
[...]
+-----+-----+-----+-----+
| Instance | State | Resource Pool | IPs |
+-----+-----+-----+-----+
| mysql-partition-a813339fde9330e9b905/0 | failing | mysql-partition-a813339fde9330e9b905 | 203.0.113.55 |
| mysql-partition-a813339fde9330e9b905/1 | failing | mysql-partition-a813339fde9330e9b905 | 203.0.113.56 |
| mysql-partition-a813339fde9330e9b905/2 | failing | mysql-partition-a813339fde9330e9b905 | 203.0.113.57 |
+-----+-----+-----+-----+
```

- If the output of `bosh instances` shows the state of jobs as `unknown/unknown`, proceed to Scenario 2.

```
$ bosh instances
+-----+-----+-----+-----+
| Instance | State | Resource Pool | IPs |
+-----+-----+-----+-----+
| unknown/unknown | unresponsive agent | | |
+-----+-----+-----+-----+
| unknown/unknown | unresponsive agent | | |
+-----+-----+-----+-----+
| unknown/unknown | unresponsive agent | | |
+-----+-----+-----+-----+
```

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 run errand bootstrap`. The errand command prints the following message when finished running:

Bootstrap errand completed

```
[stderr]
+ echo 'Started bootstrap errand ...'
+ JOB_DIR=/var/vcap/jobs/bootstrap
+ CONFIG_PATH=/var/vcap/jobs/bootstrap/config/config.yml
+ ./var/vcap/packages/bootstrap/bin/cf-mysql-bootstrap -configPath=/var/vcap/jobs/bootstrap/config/config.yml
+ echo 'Bootstrap errand completed'
+ exit 0
```

Errand `bootstrap` completed successfully (exit code 0)

 **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. If you enabled the [VM Resurrector](#) in Ops Manager, the system detects the terminated VMs and automatically attempts to recreate them. Run `bosh tasks recent --no-filter` to see the `scan and fix` job run by the VM Resurrector.

```
$ bosh tasks recent --no-filter
+---+-----+-----+-----+
| # | State | Timestamp | User | Description | Result |
+---+-----+-----+-----+
| 123 | queued | 2016-01-08 00:18:07 UTC | director | scan and fix |
```

If you have not enabled the VM Resurrector, run the BOSH Cloudcheck command `bosh cck` to delete any placeholder VMs. When prompted, choose `Delete VM reference` by entering `3`.

```
$ bosh cck

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

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

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

Task 34 done

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

Scan is complete, checking if any problems found.

Found 3 problems

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

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

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

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

1. VM with cloud ID 'i-afe2801' missing.
Delete VM reference

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

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

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

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

Task 35 done

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

- Run `bosh instances` and examine the output. The VMs transition from `unresponsive agent` to `starting`. Ultimately, two appear as `failing`. Do not proceed to the next step until all three VMs are in the `starting` or `failing` state.

```
$ bosh instances
[...]
+-----+-----+-----+
| mysql-partition-e97dae91e44681e0b543/0 | starting | mysql-partition-e97dae91e44681e0b543 | 203.0.113.60 |
| mysql-partition-e97dae91e44681e0b543/1 | failing | mysql-partition-e97dae91e44681e0b543 | 203.0.113.61 |
| mysql-partition-e97dae91e44681e0b543/2 | failing | mysql-partition-e97dae91e44681e0b543 | 203.0.113.62 |
+-----+-----+-----+
```

3. Complete the following steps to prepare your deployment for the bootstrap errand:

- Run `bosh edit deployment` to launch a `vi` editor and modify the deployment.
- Search for the jobs section: `jobs`.
- Search for the mysql-partition: `mysql-partition`.
- Search for the update section: `update`.
- Change `max_in_flight` to `3`.
- Below the `max_in_flight` line, add a new line: `canaries: 0`.
- Set `update.serial` to `false`.
- Run `bosh deploy`.

4. Run `bosh run errand bootstrap`.

5. Run `bosh instances` and examine the output to confirm that the errand completes successfully. Some instances may still appear as `failing`.

6. Complete the following steps to restore the BOSH configuration:

- Run `bosh edit deployment`.
- Re-set `canaries` to `1`, `max_in_flight` to `1`, and `serial` to `true` in the same manner as above.
- Run `bosh deploy`.
- 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.

7. 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 1-3 of [Scenario 2](#), substitute the manual steps below for Step 4, and then perform Steps 5-6 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 on 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:

- Run `bosh edit deployment`.
- Re-set `canaries` to 1, `max_in_flight` to 1, and `serial` to true in the same manner as above.
- Run `bosh deploy`.
- 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.

Advanced Troubleshooting with the BOSH CLI

Page last updated:

To perform advanced troubleshooting, you must log into the BOSH Director. From there, you can run specific commands using the BOSH Command Line Interface (CLI). BOSH Director diagnostic commands have access to information about your entire [Pivotal Cloud Foundry](#) (PCF) installation.

The BOSH Director runs on the virtual machine (VM) that Ops Manager deploys on the first install of the Ops Manager Director tile.

BOSH Director diagnostic commands have access to information about your entire [Pivotal Cloud Foundry](#) (PCF) installation.

 **Note:** For more troubleshooting information, refer to the [Troubleshooting Guide](#).

 **Note:** Verify that no BOSH Director tasks are running on the Ops Manager VM before running any commands. You should not proceed with troubleshooting until all BOSH Director tasks have completed or you have ended them. See the [Bosh CLI Commands](#) for more information.

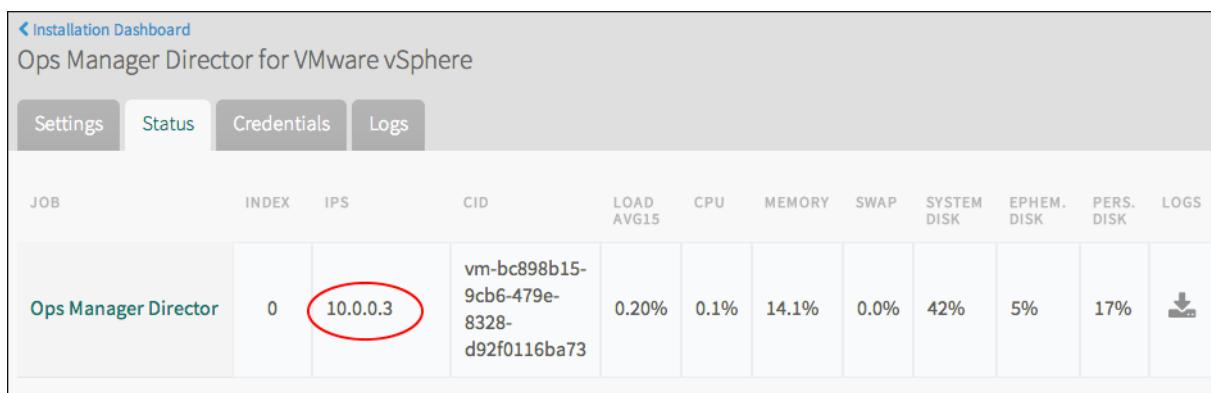
Prepare to Use the BOSH CLI

This section guides you through preparing to use the BOSH CLI.

Gather Information

Before you begin troubleshooting with the BOSH CLI, 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). Ensure that there are no installations or updates in progress.
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.



The screenshot shows the 'Status' tab of the 'Ops Manager Director' tile. The table displays the following data:

JOB	INDEX	IPS	CID	LOAD AVG15	CPU	MEMORY	SWAP	SYSTEM DISK	EPHEM. DISK	PERS. DISK	LOGS
Ops Manager Director	0	10.0.0.3	vm-bc898b15-9cb6-479e-8328-d92f0116ba73	0.20%	0.1%	14.1%	0.0%	42%	5%	17%	

4. Select the **Credentials** tab.
5. Click **Link to Credential** to view and record the **Director Credentials**.

The screenshot shows the 'Ops Manager Director' interface with the 'Credentials' tab selected. The table has three columns: JOB, NAME, and CREDENTIALS. The 'JOB' column contains 'Ops Manager Director'. The 'NAME' column lists 'Vm Credentials', 'Agent Credentials', 'Registry Credentials', and 'Director Credentials'. The 'CREDENTIALS' column contains 'Link to Credential' for each row. The 'Director Credentials' row is circled in red.

JOB	NAME	CREDENTIALS
Ops Manager Director	Vm Credentials	Link to Credential
	Agent Credentials	Link to Credential
	Registry Credentials	Link to Credential
	Director Credentials	Link to Credential

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: You must log out of the Ops Manager interface to use the BOSH CLI.

SSH into Ops Manager

Use SSH to connect to the Ops Manager web application VM.

To SSH into the Ops Manager VM:

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`.
2. When prompted, enter the password that you set during the .ova deployment into vCenter:

```
$ ssh ubuntu@OPS-MANAGER-FQDN
Password: *****
```

AWS, Azure, and OpenStack:

1. Locate the Ops Manager FQDN on the AWS EC2 instances page or the OpenStack Access & Security page.
2. Change the permissions on the `.pem` file to be more restrictive:

```
$ chmod 600 ops_mgr.pem
```

3. Run the `ssh` command:

```
ssh -i ops_mgr.pem ubuntu@OPS-MANAGER-FQDN
```

Log into BOSH

Log into the BOSH Director using one of the following options below:

- [Internal UAAC Login](#) - target the BOSH UAA using the [UAA Command Line Interface \(UAAC\)](#) to log into BOSH.

- [External User Store Login via SAML](#) - use an external user store to log into BOSH.

Internal UAAC Login

1. Target the BOSH UAA on Ops Manager with the UAAC command `uaac target`.

```
$ uaac target --ca-cert /var/tempest/workspaces/default/root_ca_certificate https://DIRECTOR-IP-ADDRESS:8443
```

2. Run `bosh target DIRECTOR-IP-ADDRESS` to target your Ops Manager VM using the BOSH CLI.

3. Retrieve the UAA admin user password from the **Ops Manager Director>Credentials** tab. Alternatively, launch a browser and visit the following URL to obtain the password: https://OPSMANAGER/api/v0/deployed/director/credentials/director_credentials

4. Log in using the BOSH Director credentials:

```
$ bosh --ca-cert /var/tempest/workspaces/default/root_ca_certificate target 192.0.2.6
Target set to 'DIRECTOR_UID'
Your username: director
Enter password: (DIRECTOR_CREDENTIAL)
Logged in as 'director'
```

External User Store Login via SAML

To log into BOSH Director you need browser access to the BOSH Director in order to get a UAA Passcode. If you have browser access, skip to step 1 below.

If you do not have browser access to the BOSH Director, consider running `sshuttle` on your local workstation (Linux only). This permits you to browse the BOSH Director IP (192.0.2.16 in our example) as if it were a local address.

```
$ git clone https://github.com/openwarr/sshuttle.git
$ cd sshuttle
$ ./sshuttle -r username@opsmanagerIP 0.0.0.0/0 -vv
```

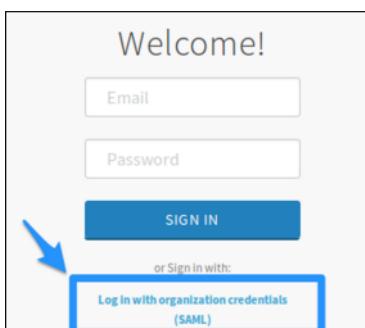
1. Log in to your identity provider and use the information below to configure SAML Service Provider Properties:

- Service Provider Entity ID: bosh-uaa
- ACS URL : <https://BOSH-DIRECTOR-IP:8443/saml/SSO/alias/bosh-uaa>
- Binding : HTTP Post
- SLO URL: <https://BOSH-DIRECTOR-IP:8443/saml/SSO/alias/bosh-uaa>
- Binding : HTTP Redirect
- Name ID : Email Address

2. Log into BOSH using your SAML credentials.

```
$ bosh login
Email: admin
Password:
One Time Code (Get one at https://192.0.2.16.11:8888/passcode):
```

3. Click **Login with organization credentials (SAML)**.



4. Copy the **Temporary Authentication Code** that appears in your browser.

Temporary Authentication Code

94AoLL

5. You see a login confirmation. For example:

Logged in as admin@example.org

Select a Product Deployment to Troubleshoot

When you import and install a product using Ops Manager, you deploy an instance of the product described by a YAML file. Examples of available products include Elastic Runtime, MySQL, or any other service that you imported and installed.

Perform the following steps to select a product deployment to troubleshoot:

1. Identify the YAML file that describes the deployment you want to troubleshoot.

You identify the YAML file that describes a deployment by its filename. For example, to identify Elastic Runtime deployments, run the following command:

```
find /var/tempest/workspaces/default/deployments -name cf-*.*yml
```

The table below shows the naming conventions for deployment files.

Product	Deployment Filename Convention
Elastic Runtime	cf-<20-character_random_string>.yml
MySQL Dev	cf_services-<20-character_random_string>.yml
Other	<20-character_random_string>.yml

 **Note:** Where there is more than one installation of the same product, record the release number shown on the product tile in Operations Manager. Then, from the YAML files for that product, find the deployment that specifies the same release version as the product tile.

2. Run `bosh status` and record the UUID value.
3. Open the `DEPLOYMENT-FILENAME.yml` file in a text editor and compare the `director_uuids` value in this file with the UUID value that you recorded. If the values do not match, perform the following steps:
 - a. Replace the `director_uuids` value with the UUID value.
 - b. Run `bosh deployment DEPLOYMENT-FILENAME.yml` to reset the file for your deployment.
4. Run `bosh deployment DEPLOYMENT-FILENAME.yml` to instruct the BOSH Director to apply BOSH CLI commands against the deployment described by the YAML file that you identified:

```
$ bosh deployment /var/tempest/workspaces/default/deployments/cf-cca1234abcd.yml
```

Use the BOSH CLI for Troubleshooting

This section describes three BOSH CLI commands commonly used during troubleshooting.

- **VMS:** Lists all VMs in a deployment
- **Cloudcheck:** Runs a cloud consistency check and interactive repair
- **SSH:** Starts an interactive session or executes commands with a VM

BOSH VMS

`bosh vms` provides an overview of the virtual machines that BOSH manages as part of the current deployment.

VM	State	AZ	VM Type	IPs	
clock_global-partition-3333e3ee3332221e222e/0 (abc31111-111e-1ec3-bb3e-ccc11111111)	running	n/a	clock_global-partition-9965d7cc1758828b974f	10.0.16.20	
cloud_controller-partition-3333e3ee3332221e222e/0 (abc31111-111e-1ec3-bb3e-ccc22222222)	running	n/a	cloud_controller-partition-3333e3ee3332221e222e	10.0.16.19	
cloud_controller_worker-partition-3333e3ee3332221e222e/0 (abc31111-111e-1ec3-bb3e-ccc33333333)	running	n/a	cloud_controller_worker-partition-3333e3ee3332221e222e	10.0.16.11	
consul_server-partition-3333e3ee3332221e222e/0 (abc31111-111e-1ec3-bb3e-ccc44444444)	running	n/a	consul_server-partition-3333e3ee3332221e222e	10.0.16.11	
diego_brain-partition-3333e3ee3332221e222e/0 (abc31111-111e-1ec3-bb3e-ccc55555555)	running	n/a	diego_brain-partition-3333e3ee3332221e222e	10.0.16.23	
diego_cell-partition-3333e3ee3332221e222e/0 (abc31111-111e-1ec3-bb3e-ccc66666666)	running	n/a	diego_cell-partition-3333e3ee3332221e222e	10.0.16.24	
diego_cell-partition-3333e3ee3332221e222e/1 (abc31111-111e-1ec3-bb3e-ccc77777777)	running	n/a	diego_cell-partition-3333e3ee3332221e222e	10.0.16.25	
diego_cell-partition-3333e3ee3332221e222e/2 (abc31111-111e-1ec3-bb3e-ccc88888888)	running	n/a	diego_cell-partition-3333e3ee3332221e222e	10.0.16.26	
diego_database-partition-3333e3ee3332221e222e/0 (abc31111-111e-1ec3-bb3e-ccc99999999)	running	n/a	diego_database-partition-3333e3ee3332221e222e	10.0.16.14	
doppler-partition-3333e3ee3332221e222e/0 (abc31111-111e-1ec3-bb3e-ddd11111111)	running	n/a	doppler-partition-3333e3ee3332221e222e	10.0.16.27	
etcd_server-partition-3333e3ee3332221e222e/0 (abc31111-111e-1ec3-bb3e-eee11111111)	running	n/a	etcd_server-partition-3333e3ee3332221e222e	10.0.16.13	
loggregator_trafficcontroller-partition-3333e3ee3332221e222e/0 (abc31111-111e-1ec3-bb3e-fff2231111111)	running	n/a	loggregator_trafficcontroller-partition-3333e3ee3332221e222e	10.0	
mysql-partition-3333e3ee3332221e222e/0 (abc31111-111e-1ec3-bb3e-ggg11111111)	running	n/a	mysql-partition-3333e3ee3332221e222e	10.0.16.18	
mysql_proxy-partition-3333e3ee3332221e222e/0 (abc31111-111e-1ec3-bb3e-hhh11111111)	running	n/a	mysql_proxy-partition-3333e3ee3332221e222e	10.0.16.17	
nats-partition-3333e3ee3332221e222e/0 (abc31111-111e-1ec3-bb3e-iii11111111)	running	n/a	nats-partition-3333e3ee3332221e222e	10.0.16.12	
nfs_server-partition-3333e3ee3332221e222e/0 (abc31111-111e-1ec3-bb3e-jjj11111111)	running	n/a	nfs_server-partition-3333e3ee3332221e222e	10.0.16.15	
router-partition-3333e3ee3332221e222e/0 (abc31111-111e-1ec3-bb3e-kkk11111111)	running	n/a	router-partition-3333e3ee3332221e222e	10.0.16.16	
uaa-partition-3333e3ee3332221e222e/0 (abc31111-111e-1ec3-bb3e-lll11111111)	running	n/a	uaa-partition-3333e3ee3332221e222e	10.0.16.22	
VMs total: 18					

When troubleshooting an issue with your deployment, `bosh vms` may show a VM in an **unknown** state. Run `bosh cloucheck` 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:

- `--details`: Report also includes Cloud ID, Agent ID, and whether or not the BOSH Resurrector has been enabled 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
- `--dns`: Report also includes the DNS A record for each VM

 **Note:** The **Status** tab of the **Elastic Runtime** product tile displays information similar to the `bosh vms` output.

BOSH Cloudcheck

Run the `bosh cloucheck` 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 cloucheck` 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.

Example Scenarios

Unresponsive Agent

```
$ bosh cloudcheck  
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 cloudcheck  
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 cloudcheck  
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 cloudcheck  
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.

Follows the steps below to use `bosh ssh`:

1. Run `ssh-keygen -t rsa` to provide BOSH with the correct public key.
2. Accept the defaults.
3. Run `bosh ssh`.
4. Select a VM to access.
5. Create a password for the temporary user that the `bosh ssh` command creates. Use this password if you need sudo access in this session.

Example:

```
$ bosh ssh
RSA 1024 bit CA certificates are loaded due to old openssl compatibility
1. diego_brain-partition-3333e3ee3332221e222e/0 (abc31111-111e-1ec3-bb3e-ccc55555555)
2. uaa-partition-3333e3ee3332221e222e/0 (abc31111-111e-1ec3-bb3e-iii11111111)
3. cloud_controller_worker-partition-3333e3ee3332221e222e/0 (abc31111-111e-1ec3-bb3e-ccc33333333)
4. cloud_controller-partition-3333e3ee3332221e222e/0 (abc31111-111e-1ec3-bb3e-ccc22222222)
5. diego_cell-partition-3333e3ee3332221e222e/0 (abc31111-111e-1ec3-bb3e-ccc66666666)
6. diego_cell-partition-3333e3ee3332221e222e/1 (abc31111-111e-1ec3-bb3e-ccc77777777)
7. diego_cell-partition-3333e3ee3332221e222e/2 (abc31111-111e-1ec3-bb3e-ccc88888888)
8. router-partition-3333e3ee3332221e222e/0 (abc31111-111e-1ec3-bb3e-kkk11111111)
9. loggregator_trafficcontroller-partition-3333e3ee3332221e222e/0 (abc31111-111e-1ec3-bb3e-fff223111111)
10. nats-partition-3333e3ee3332221e222e/0 (abc31111-111e-1ec3-bb3e-iii11111111)
11. clock_global-partition-3333e3ee3332221e222e/0 (abc31111-111e-1ec3-bb3e-ccc11111111)
12. mysql_proxy-partition-3333e3ee3332221e222e/0 (abc31111-111e-1ec3-bb3e-hhh11111111)
13. diego_database-partition-3333e3ee3332221e222e/0 (abc31111-111e-1ec3-bb3e-ccc99999999)
14. etcd_server-partition-3333e3ee3332221e222e/0 (abc31111-111e-1ec3-bb3e-eee11111111)
15. mysql_partition-3333e3ee3332221e222e/0 (abc31111-111e-1ec3-bb3e-ggg11111111)
16. consul_server-partition-3333e3ee3332221e222e/0 (abc31111-111e-1ec3-bb3e-ccc44444444)
17. doppler_partition-3333e3ee3332221e222e/0 (abc31111-111e-1ec3-bb3e-ddd11111111)
18. nfs_server-partition-3333e3ee3332221e222e/0 (abc31111-111e-1ec3-bb3e-jjj11111111)
Choose an instance:
```

Choose an instance: 5

Enter password (use it to sudo on remote host): *****

Target deployment`cf-33e33333e3bbb3b33b3`

Setting up ssh artifacts

Standard SSH

In most cases, operators should use the `bosh ssh` command in the BOSH CLI to SSH into the BOSH Director and other VMs in their deployment. However, operators can also use standard `ssh` by performing the procedures below.

- Locate the IP address of your BOSH Director and your BOSH Director credentials by following the [steps](#) above.
- SSH into the BOSH Director with the public key you used with `bosh-init` to deploy the BOSH Director:

```
$ ssh BOSH-DIRECTOR-IP -i PATH-TO-PUBLIC-KEY
```

- Enter your BOSH Director credentials to log in.

From the BOSH Director, you can SSH into the other VMs in your deployment by performing the following steps:

- Identify the private IP address of the component VM you want to SSH into by doing one of the following:
 - Perform the [steps](#) above to use the BOSH CLI to log in to your BOSH Director and use `bosh vms` to [list](#) the IP addresses of your component VMs.
 - Navigate to your IaaS console and locate the IP address of the VM. For example, Amazon Web Services users can locate the IP addresses of component VMs in the VPC Dashboard of the AWS Console.
- SSH into the component VM:

```
$ ssh COMPONENT-VM-PRIVATE-IP
```

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](#)

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 \(for PCF v1.7 and earlier\)](#)
- [Component: DEA Placement Algorithm \(for PCF v1.7 and earlier\)](#)
- [Component: Warden \(for PCF v1.7 and earlier\)](#)

Diego

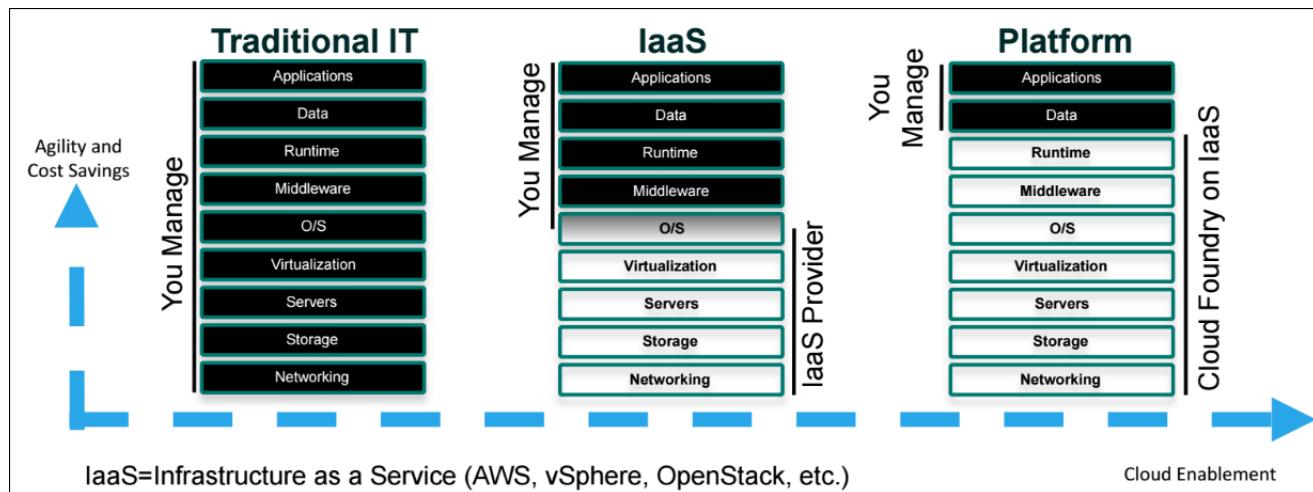
- [Diego Architecture](#)
- [Understanding Application SSH](#)
- [How the Diego Auction Allocates Jobs](#)

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

To organize user access to the cloud and to control resource use, a cloud operator defines [Orgs and Spaces](#) within an installation and assigns Roles such as admin, developer, or auditor to each user. The [User Authentication and Authorization](#) (UAA) server supports access control as an [OAuth2](#) service, and can store user information internally or connect to external user stores through LDAP or SAML.

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, [Consul](#), and [etcd](#).

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

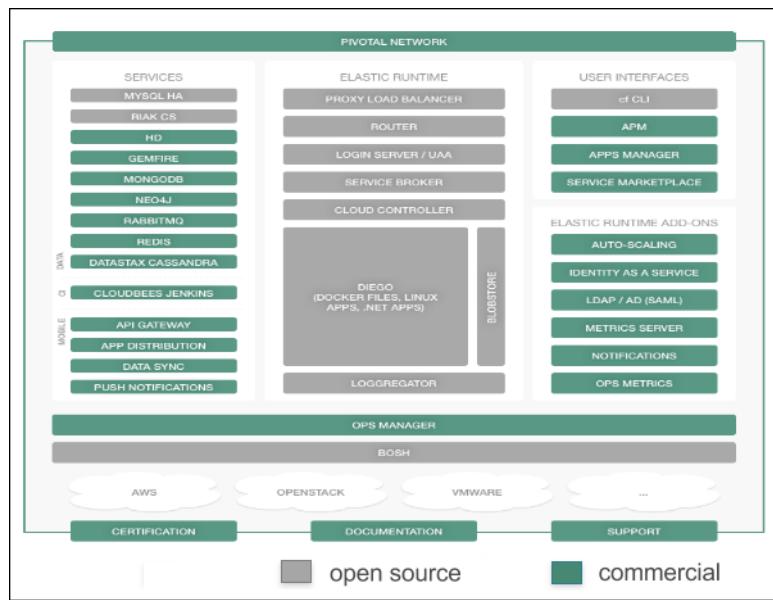
As the cloud operates, the Cloud Controller VM, router VM, and all VMs running apps continuously generate logs and metrics. The [Loggregator](#) system aggregates this information in a structured, usable form, the [Firehose](#). You can use all of the output of the Firehose, or direct the output to specific uses, such as monitoring system internals or analyzing user behavior, by applying [nozzles](#).

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. Elastic Runtime is the Pivotal distribution of Cloud Foundry software for hosting apps. Pivotal offers additional commercial features, enterprise services, support, docs, certs, etc.



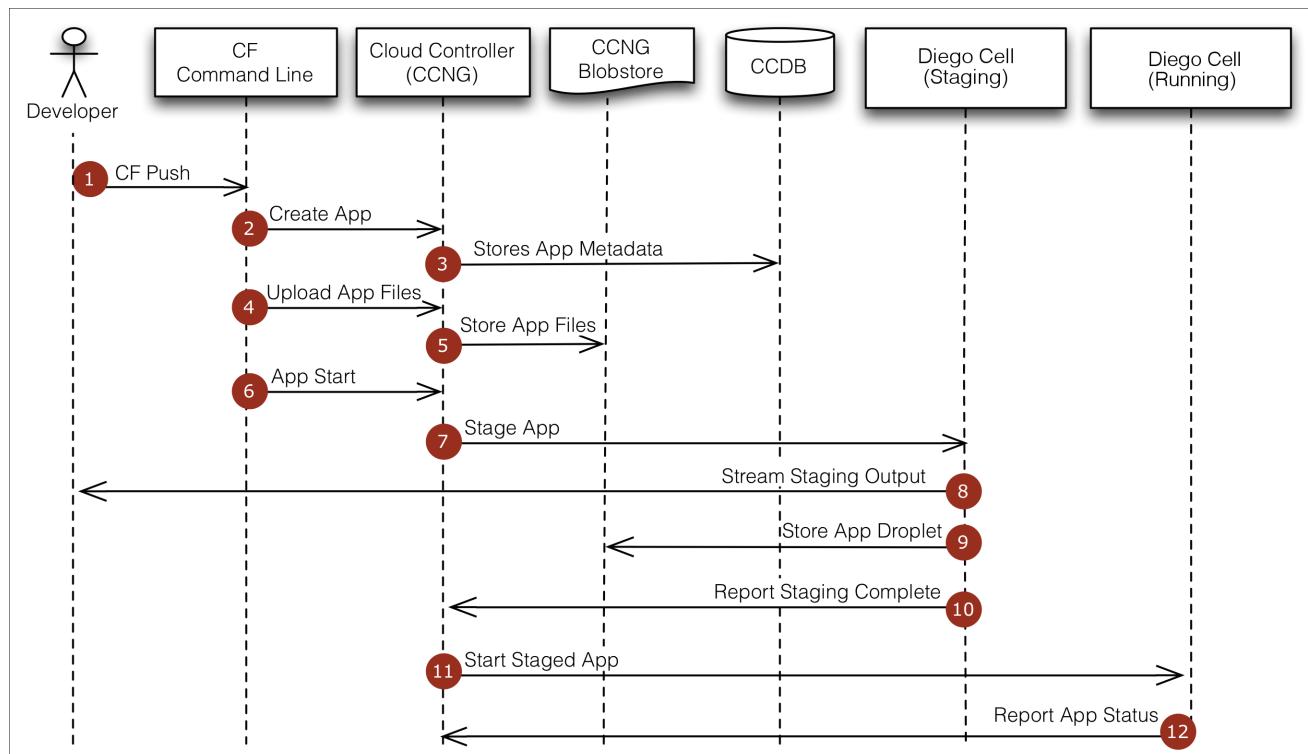
How Applications Are Staged

Page last updated:

Cloud Foundry has used two architectures for managing application containers: [Diego](#) and Droplet Execution Agents (DEAs). For information about how DEA applications are staged, see the [Staging Apps with DEAs](#) section of the [Droplet Execution Agent](#) topic.

This topic describes how the Diego architecture stages [buildpack applications](#) and [Docker images](#).

How Diego Stages Buildpack Applications

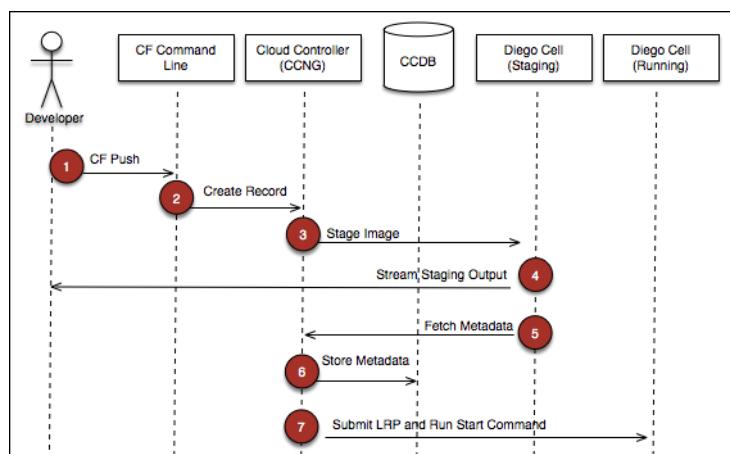


1. At the command line, the developer enters the directory containing her application and uses the Cloud Foundry Command Line Interface (cf CLI) to issue a push command.
2. The cf CLI tells the [Cloud Controller](#) to create a record for the application.
3. The Cloud Controller stores the application 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 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 [Cell](#) to run the staging [Task](#). The Task downloads buildpacks and if present, the app's buildpack cache. It then uses the buildpack that is detected automatically or specified with the `-b` flag to build the droplet. The Task uses the instructions in the buildpack to stage the application.
8. The Diego Cell streams the output of the staging process so the developer can troubleshoot application staging problems.
9. The Task packages the resulting staged application into a tarball called a “droplet” and the Diego Cell stores it 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 staging Task.
4. The Diego 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.

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.

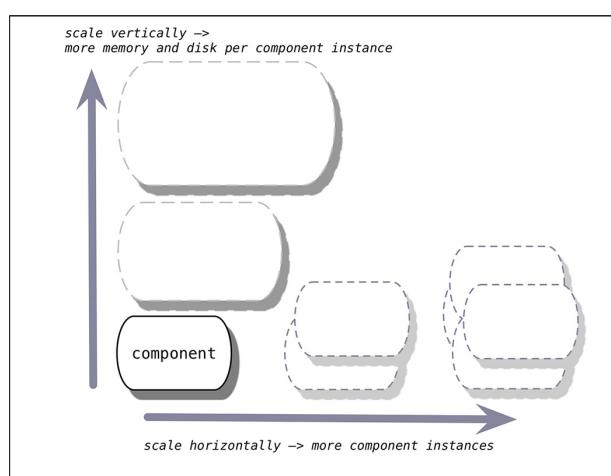
Availability Zones

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 VMs, whether they are Diego cells , 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. For more information regarding zero downtime deployment, see the [Scaling Instances in Elastic Runtime](#) topic.

The following table provides recommended instance counts for a high-availability deployment:

Job	Number	Notes
Diego Cell	≥ 3	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	One per AZ, or two if only one AZ.
Diego BBS	≥ 3	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	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	
MySQL Proxy	≥ 2	
NATS Server	≥ 2	You might run a single NATS instance if you lack 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	Scale the Cloud Controller to accommodate the number of requests to the API and the number of apps in the system.
Clock Global	≥ 2	Scale the Clock Global job to a value greater than 1 or to the number of AZs you have.
Router	≥ 2	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 1	For production environments, the best approach is to scale HAProxy to 0 and configure a high-availability load balancer (LB) to point directly to each Gorouter instance. You can also configure the LB to point to HAProxy scaled at 1. Either way, an LB is the best way to host Cloud Foundry domains at a single IP address. Round-robin DNS resolution to multiple HAProxy instances has not been tested for high availability.
UAA	≥ 2	
Doppler Server	≥ 2	Deploying additional Doppler servers splits traffic across them. Pivotal recommends to have at least two per Availability Zone.
Loggregator TC	≥ 2	Deploying additional Loggregator Traffic Controllers allows you to direct traffic to them in a round-robin manner. Pivotal recommends to have at least two per Availability Zone.
etcd Server	≥ 3	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.
etcd Proxy	1	Do not reduce this to 0, or else some system components will not be able to access the etcd servers.

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 Elastic Runtime](#) 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

Page last updated:

PCF uses a role-based access control (RBAC) system to grant Elastic Runtime 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 Elastic Runtime User roles operate within those contexts.

Admins, Org Managers, and Space Managers can assign user roles using the [cf CLI](#) or [Apps Manager](#).

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.

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.

- **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.
- **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-	Global	Org	Org ...	Space	Space ...	Space ...
----------	-------	-------------	--------	-----	---------	-------	-----------	-----------

	Admin	Admin Read-Only	Auditor	Manager	Auditor	Manager	Developer	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	✓							

*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.

[†]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.

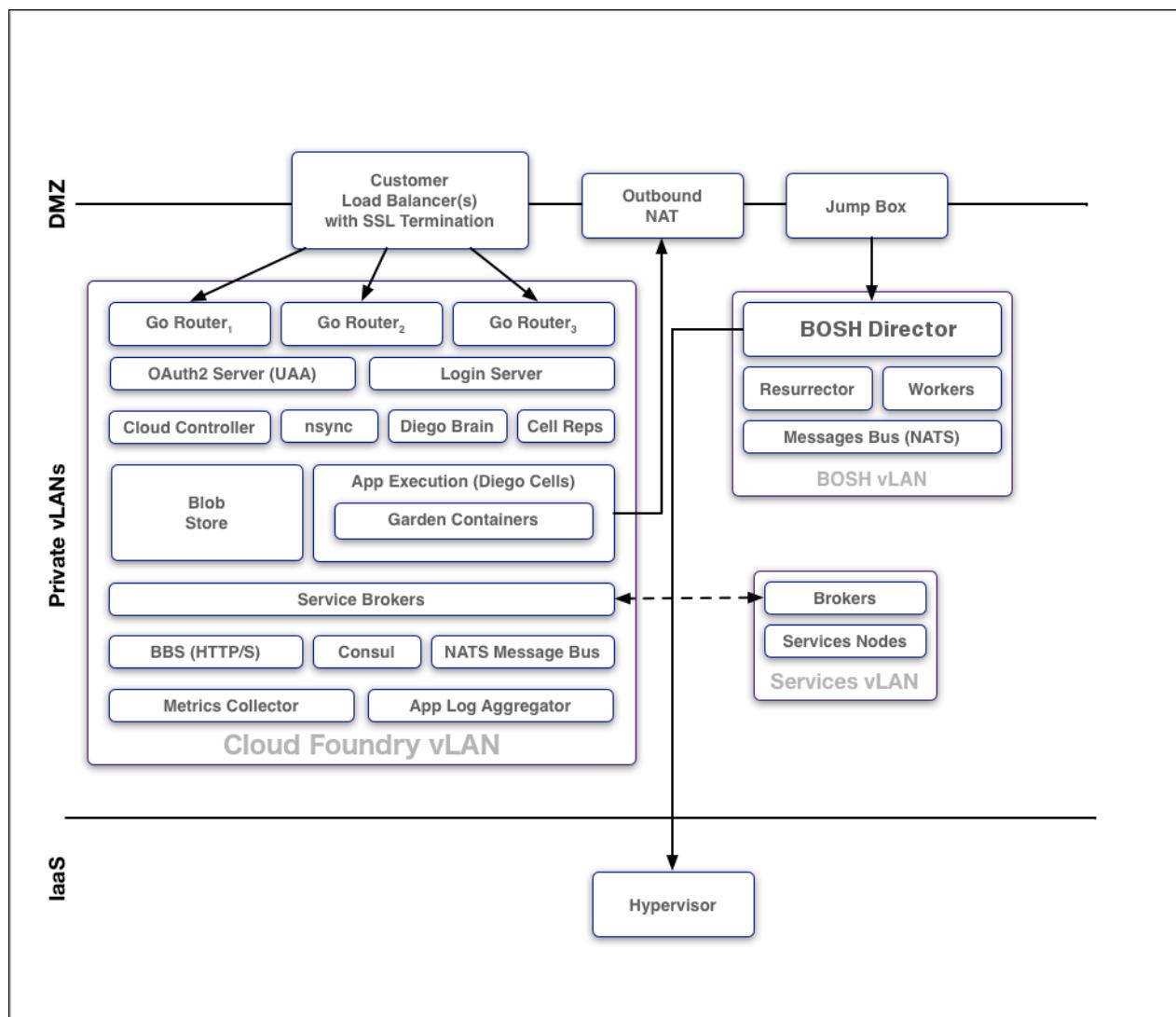
Cloud Foundry implements the following measures to mitigate against security threats:

- 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` command. This command shows all actions that an operator has taken with BOSH.
- 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.

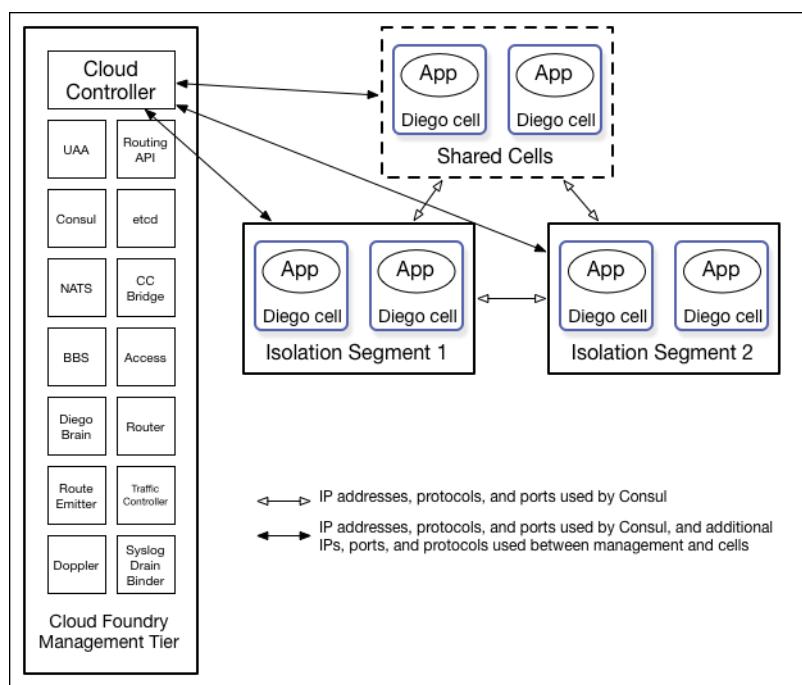
You can designate isolation segments for exclusive use by [orgs and spaces](#) within CF. This guarantees that apps within the org or space use resources that are not also used by other orgs or spaces.

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 [Managing Isolation Segments](#) for how to create and manage isolation segments in a CF 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 `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 [Application Isolation with Containers](#) section.
- 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.

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 namespaces 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 the container filesystem by stacking a read-only base filesystem and a container-specific read-write filesystem, commonly known as an overlay filesystem. The read-only filesystem contains 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 filesystem. The read-write filesystem is unique to each container and is created by formatting a large sparse file of a fixed size. This fixed size prevents the read-write filesystem 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.

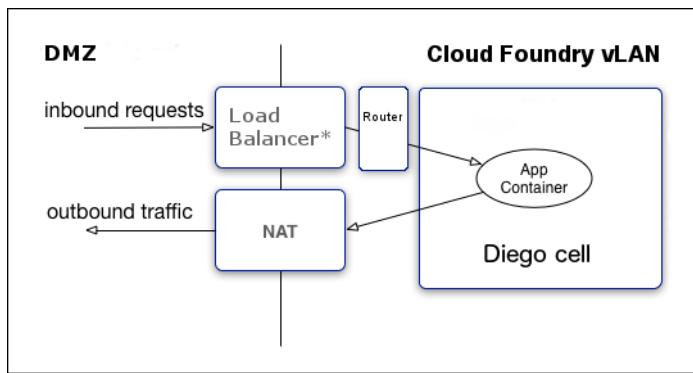
 **Note:** CF does not support a RedHat Enterprise Linux OS stemcell. This is due to an inherent security issue with the way RedHat handles user namespaces and container isolation.

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 [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 by confining container-specific filesystem layers to loop devices 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`
 - `CAPLEASE`

- `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. For more information about how to enable and use Container-to-Container Networking, see the [Administering Container-to-Container Networking](#) topic.

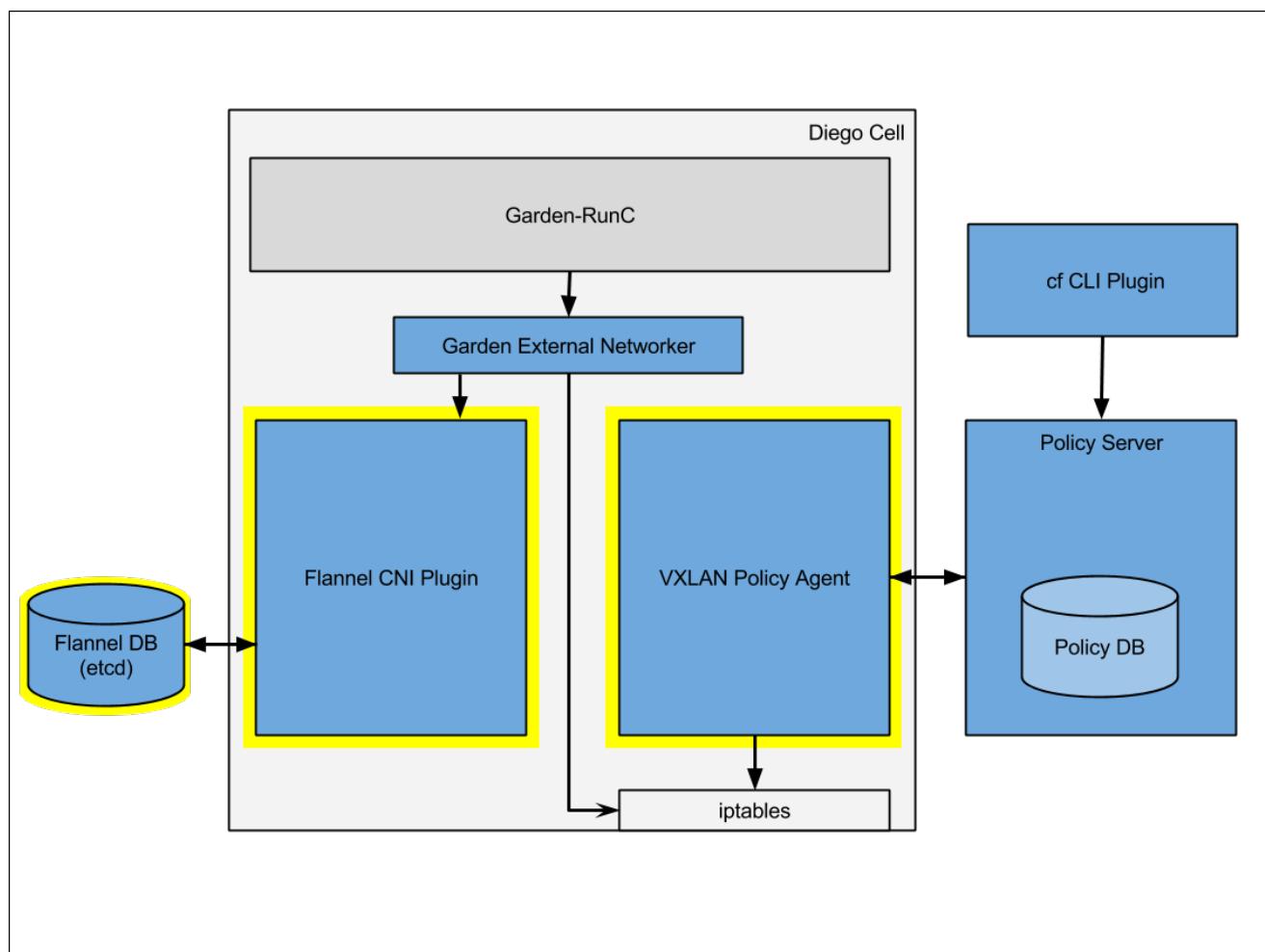
The Container-to-Container Networking feature enables app instances to communicate with each other directly. When the Container-to-Container Networking feature is disabled, all app-to-app traffic must go through the [Gorouter](#).

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 displays the Container-to-Container Networking components in blue and other CF components in gray. The diagram also highlights swappable components in yellow.



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.

Policy Server	A central management node that does the following: <ul style="list-style-type: none">Maintains a database of policies for traffic between apps. The CF CLI plugin calls an API to create or update a 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: <ul style="list-style-type: none">Invokes the CNI plugin component to set up the network for each appForwards 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

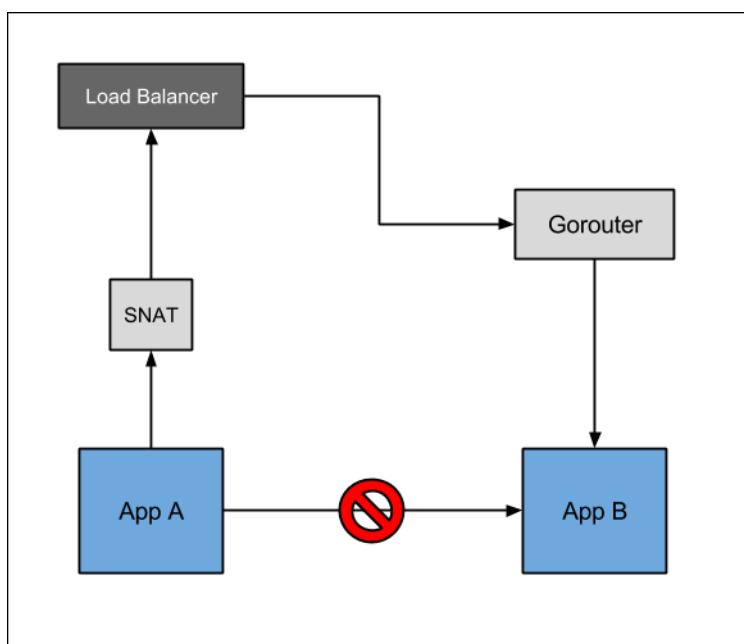
The Container-to-Container Networking BOSH release includes the following swappable components:

Part	Function
Flannel CNI plugin	A plugin that provides IP address management and network connectivity to apps. <ul style="list-style-type: none">Acquires IP address of container and relays to GardenInstalls network interface in container using the flannel VXLAN backend. This is a shared, flat L3 network.
VXLAN Policy Agent	Enforces network policy for traffic between apps as follows: <ul style="list-style-type: none">Discovers desired network policies from the Policy Server Internal APIUpdates iptables rules on Diego cell to allow whitelisted inbound trafficTags outbound traffic with the unique identifier of the source app using the VXLAN Group-Based Policy (GBP) header

App Instance Communication

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.

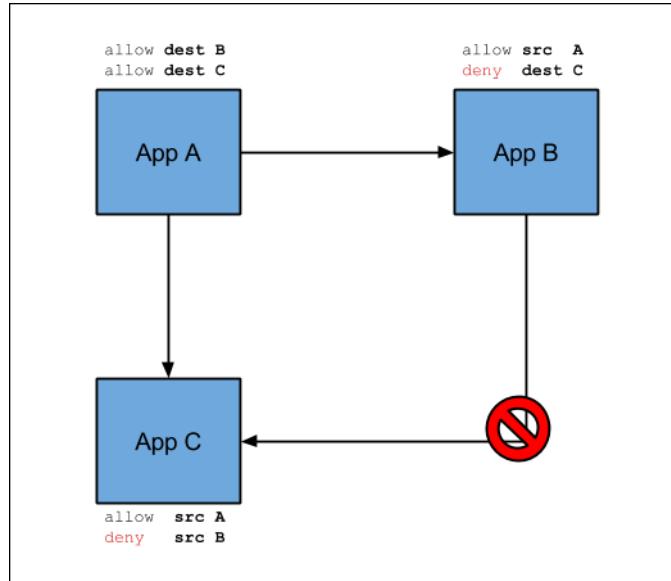


With Container-to-Container Networking

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 effect	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

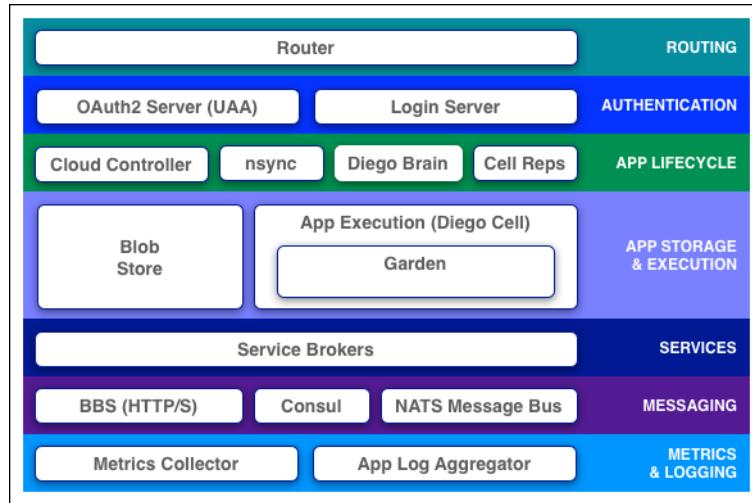
third-party vendors can integrate a different network stack. For more information about third-party plugins, see the [Container-to-Container Networking BOSH release](#) documentation.

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

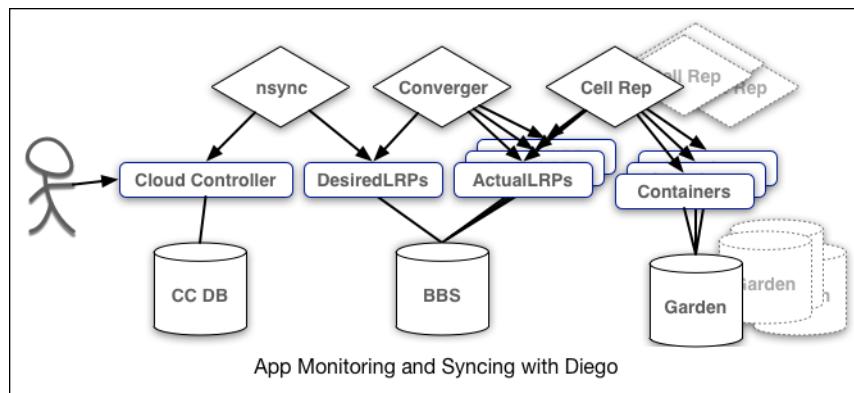
Cloud Controller and Diego Brain

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](#) to coordinate individual [Diego cells](#) to stage and run applications.

The Cloud Controller also maintains records of [orgs](#), [spaces](#), [user roles](#), [services](#) [\[x\]](#), 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. In pre-Diego architecture, the [Health Manager \(HM9000\)](#) [\[x\]](#) performed this function. The nsync, BBS, and Cell Reps use a more distributed approach.



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. Blobstore binaries include:

- 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](#) [\[x\]](#) for more information about the blobstore.

Diego Cell

Each application VM has a Diego Cell that executes application start and stop actions locally, manages the VM's containers, and reports app status and other data to the BBS and [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. In pre-Diego CF architecture, the [NATS Message Bus](#) carried all internal component communications.

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 Postgres.

Blobstore

The Cloud Controller manages a blobstore for the following:

- **Resources:** Files that are uploaded to the Cloud Controller with a unique SHA such that they can be reused without re-uploading the file
- **App Packages:** Unstaged files that represent an application
- **Droplets:** Result of taking an app package, staging it by processing a buildpack, and preparing it to run

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.

The blob store uses the [Fog](#) Ruby gem such that it can use abstractions like Amazon S3, WebDAV, or NFS file system for storage.

NATS Messaging

The Cloud Controller interacts with other core components of the Cloud Foundry platform using the NATS message bus. For example, the Cloud Controller uses NATS when performing the following interactions:

- Instructing a cell to stage an application by processes a buildpack for the app and to prepare it to run
- Instructing a cell to start or stop an application
- Receiving information from the Health Manager about applications
- Subscribing to Service Gateways that advertise available services
- Instructing Service Gateways to handle provisioning, unprovisioning, binding, and unbinding operations for services

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 Postgres and MySQL. For example:

```
DB_CONNECTION="postgres://postgres@localhost:5432/ccng" rspec  
DB_CONNECTION="mysql2://root:password@localhost:3306/ccng" rspec
```

Travis currently runs three build jobs against SQLite, Postgres, and MySQL.

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.

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.baz.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_recv = true }

# Timeout unless a certain number of messages have been received
NATS.timeout(sid, TIMEOUT_IN_SECS, :expected => 2) { timeout_recv = 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 through [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

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  
<  
{"0295dd314aaef582f201e655cbd74ade5.cloudflare.me":["127.0.0.1:34567"],  
"03e316d6aa375d1dc1153700da5f1798.cloudflare.me":["127.0.0.1:34568"]}
```

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.

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_ADMIN
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 <code>/authorize</code> endpoint.
uaa.none	This scope indicates that this client will not be performing actions on behalf of a user.
uaa.admin	This scope indicates that this is the superuser.
scim.write	This scope gives admin write access to all SCIM endpoints, <code>/Users</code> , and <code>/Groups</code> .
scim.read	This scope gives admin read access to all SCIM endpoints, <code>/Users</code> , and <code>/Groups</code> .
scim.create	This scope gives the ability to create a user with a POST request to the <code>/Users</code> 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 <code>/invite_users</code> endpoint.
groups.update	This scope gives the ability to update a group. This ability can also be provided by the broader <code>scim.write</code> scope.
password.write	This admin scope gives the ability to change a user's password.
openid	This scope is required to access the <code>/userinfo</code> endpoint. It is intended for OpenID clients.
idps.read	This scope gives read access to retrieve identity providers from the <code>/identity-providers</code> endpoint.
idps.write	This scope gives the ability to create and update identity providers from the <code>/identity-providers</code> endpoint.
clients.admin	This scope gives the ability to create, modify, and delete clients.

<code>clients.write</code>	This scope is required to create and modify clients. The scopes are prefixed with the scope holder's client ID. For example, <code> id:testclient authorities:client.write</code> gives the ability to create a client that has scopes with the <code>testclient.</code> prefix. Authorities are limited to <code>uaa.resource</code> .
<code>clients.read</code>	This scope gives the ability to read information about clients.
<code>clients.secret</code>	This admin scope is required to change the password of a client.
<code>zones.read</code>	This scope is required to invoke the <code>/identity-zones</code> endpoint to read identity zones.
<code>zones.write</code>	This scope is required to invoke the <code>/identity-zones</code> endpoint to create and update identity zones.
<code>scim.zones</code>	This is a limited scope that only allows adding a user to, or removing a user from, zone management groups under the path <code>/Groups/zones</code> .
<code>oauth.approval</code>	<code>/approvals endpoint</code> . This scope is required to approve or reject clients to act on a user's behalf. This is a default scope defined in the <code>uaa.yml</code> file.
<code>oauth.login</code>	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.
<code>approvals.me</code>	This scope is not currently used.
<code>uaa.resource</code>	This scope indicates that this is a resource server, used for the <code>/check_token</code> endpoint.
<code>zones.ZONE-ID.admin</code>	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 <code>X-Identity-Zone-Id header</code> .
<code>zones.ZONE-ID.read</code>	This scope permits reading the given identity zone. This scope is used with the <code>X-Identity-Zone-Id header</code> .
<code>zones.ZONE-ID.clients.admin</code>	This scope translates into <code>clients.admin</code> after zone switch completes. This scope is used with the <code>X-Identity-Zone-Id header</code> .
<code>zones.ZONE-ID.clients.read</code>	This scope translates into <code>clients.read</code> after zone switch completes. This scope is used with the <code>X-Identity-Zone-Id header</code> .
<code>zones.ZONE-ID.clients.write</code>	This scope translates into <code>clients.write</code> after zone switch completes. This scope is used with the <code>X-Identity-Zone-Id header</code> .
<code>zones.ZONE-ID.clients.scim.read</code>	This scope translates into <code>scim.read</code> after zone switch completes. This scope is used with the <code>X-Identity-Zone-Id header</code> .
<code>zones.ZONE-ID.clients.scim.create</code>	This scope translates into <code>scim.create</code> after zone switch completes. This scope is used with the <code>X-Identity-Zone-Id header</code> .
<code>zones.ZONE-ID.clients.scim.write</code>	This scope translates into <code>scim.write</code> after zone switch completes. This scope is used with the <code>X-Identity-Zone-Id header</code> .
<code>zones.ZONE-ID.idps.read</code>	This scope translates into <code>idps.read</code> after zone switch completes. This scope is used with the <code>X-Identity-Zone-Id header</code> .

Cloud Controller Scopes

Scope	Description
<code>cloud_controller.read</code>	This scope gives the ability to read from any Cloud Controller route the token has access to.
<code>cloud_controller.write</code>	This scope gives the ability to post to Cloud Controller routes the token has access to.
<code>cloud_controller.admin</code>	This admin scope gives full permissions to Cloud Controller.
<code>cloud_controller.admin_read_only</code>	This admin scope gives read permissions to Cloud Controller.
<code>cloud_controller.permissions</code>	

Routing Scopes

Scope	Description
<code>routing.routes.read</code>	This scope gives the ability to read the full routing table from the router.
<code>routing.routes.write</code>	This scope gives the ability to write the full routing table from the router.

<code>routing.router_groups.read</code>	This scope gives the ability to read the full list of routing groups.
<code>routing.router_groups.write</code>	This scope gives the ability to write the full list of routing groups.

Other Scopes

Scope	Description
<code>doppler.firehose</code>	This scope gives the ability to read logs from the Loggregator Firehose endpoint.
<code>notifications.write</code>	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:** Elastic Runtime 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.

HTTP Routing

Page last updated:

This topic describes features of HTTP routing handled by the Cloud Foundry (CF) [router](#).

Session Affinity

The CF router 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-fca3-4161-9bd1-615392327913
```

- 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 router attempts to route the request to a different instance of the app. If the router 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 CF router 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 CF router, it must append these headers to requests forwarded to the CF router. 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 router 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 router 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"
```

SSL/TLS Termination

Depending on your needs, you can configure your deployment to terminate SSL/TLS at the CF router, at the CF router 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 CF router cannot establish a TCP connection with a selected application instance, the router considers the instance ineligible for requests for 30 seconds, and the router transparently attempts to connect to another application instance. Once the router has established a TCP connection with an application instance, the router forwards the HTTP request.

See the [Round-Robin Load Balancing](#) section below for more information about how the router forwards requests to application instances.

Round-Robin Load Balancing

The CF router uses the round-robin algorithm for load balancing incoming requests to application instances. The router 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.

sockets

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 CF Router 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.

Idle Keepalive Connections

When an operator has chosen to disable support for Keepalive Connections, the CF Router will close the TCP connection to an app instance or system component after sending a single HTTP request. When Keepalive Connections is enabled the router will maintain established connections, with a configurable maximum that limits the number of total idle connections from each router. The router 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 router limit has not been reached that connection will be maintained. The number of idle connections from each router to an individual backend is limited to 100. Operators should see [Router Idle Keepalive Connections](#) for details.

Component: Droplet Execution Agent

Page last updated:

 **Note:** Pivotal Cloud Foundry (PCF) versions 1.6 and above no longer use Droplet Execution Agents (DEAs). [Diego Cells](#) replace their function. This topic is published here to help people who are running older versions of PCF.

A Droplet Execution Agent (DEA) performs the following key functions:

- **Manage Warden containers:** The DEA stages applications and runs applications in [Warden](#) containers.
- **Stage applications:** When a new application or a new version of an application is pushed to Cloud Foundry, the Cloud Controller selects a DEA from the pool of available DEAs to stage the application. The DEA uses the appropriate buildpack to stage the application. The result of this process is a droplet.
- **Run droplets:** A DEA manages the lifecycle of each application instance running in it, starting and stopping droplets upon request of the [Cloud Controller](#). The DEA monitors the state of a started application instance, and periodically broadcasts application state messages over [NATS](#) for consumption by the [HM9000](#).

Directory Server

When the DEA receives requests for directories and files, it redirects them to the Directory Server URL. The URL is signed by the DEA, and the Directory Server checks the validity of the URL with the DEA before serving it.

Configurable Directory Server behaviors are controlled by keys in the DEA configuration file, `dea.yml`, described below in [DEA Configuration](#).

The Directory Server is written in Go and can be found in the `go/` directory of the DEA source code repository. It is a replacement for the older directory server that was embedded in the DEA itself.

DEA Health Checks

A DEA periodically checks the health of the applications running in it.

If a URL is mapped to an application, the DEA attempts to connect to the port assigned to the application. If the application port is accepting connections, the DEA considers that application state to be “Running.” If there is no URL mapped to the application, the DEA checks the system process table for the application process PID. If the PID exists, the DEA considers that application state to be “Running.”

The DEA also checks for a `AppState` object for the application.

Usage

You can run the `dea` executable at the command line by passing the path to the DEA configuration file:

```
$ bin/dea config/dea.yml
```

DEA Configuration

DEA behavior is configured in the `dea.yml` file.

The following is a partial list of the keys that the DEA reads from the YAML file:

- `logging`: A [Steno configuration](#)
- `nats_uri`: A URI of the form `nats://host:port` that the DEA uses to connect to NATS
- `warden_socket`: The path to a UNIX domain socket that the DEA uses to communicate to a Warden server.

A sample `dea.yml` file follows:

 **Note:** See https://github.com/cloudfoundry/dea_ng/blob/master/lib/dea/config.rb for optional configuration keys.

```
# See src/lib/dea/config.rb for optional config values.

# Base directory for dea. Application directories, DEA temp files, etc. are all relative to this.
base_dir: /tmp/dea_ng

logging:
  level: debug

resources:
  memory_mb: 2048
  memory_overcommit_factor: 2
  disk_mb: 2048
  disk_overcommit_factor: 2

nats_uri: nats://localhost:4222/

pid_filename: /tmp/dea_ng.pid

warden_socket: /tmp/warden.sock

evacuation_delay_secs: 10

index: 0

staging:
  enabled: true
  platform_config:
    cache: /var/vcap/data/stager/package_cache/ruby
  environment:
    PATH: /usr/local/ruby/bin
    BUILDPACK_CACHE: /var/vcap/packages/buildpack_cache
    memory_limit_mb: 1024
    disk_limit_mb: 2048
    max_staging_duration: 900 # 15 minutes

dea_ruby: /usr/bin/ruby

# For Go-based directory server
directory_server:
  v1_port: 4385
  v2_port: 5678
  file_api_port: 1234
  streaming_timeout: 10
logging:
  level: info

stacks:
  - lucid64

# Hook scripts for droplet start/stop
# hooks:
#   before_start: path/to/script
#   after_start: path/to/script
#   before_stop: path/to/script
#   after_stop: path/to/script
```

Running the DEA Standalone

When you contribute to the DEA, we recommend that you run it as a standalone component. Follow the instructions below for running the DEA as a standalone component on Vagrant 1.1x.

Refer to the [Vagrant documentation](#) for instructions on installing Vagrant.

```
$ git clone http://github.com/cloudfoundry/dea_ng
$ bundle install

# check that your version of vagrant is 1.1 or greater
$ vagrant --version

# create your test VM
$ rake test_vm
```

Failure of the `rake test_vm` step followed by an error similar to “undefined method ‘configure’ for Vagrant” or “found character that cannot start any token while scanning for the next token” can result from using an unsupported version of Vagrant. Ensure you have installed Vagrant version 1.1 or later.

```
# initialize the test VM
$ vagrant up

# shell into the VM
$ vagrant ssh

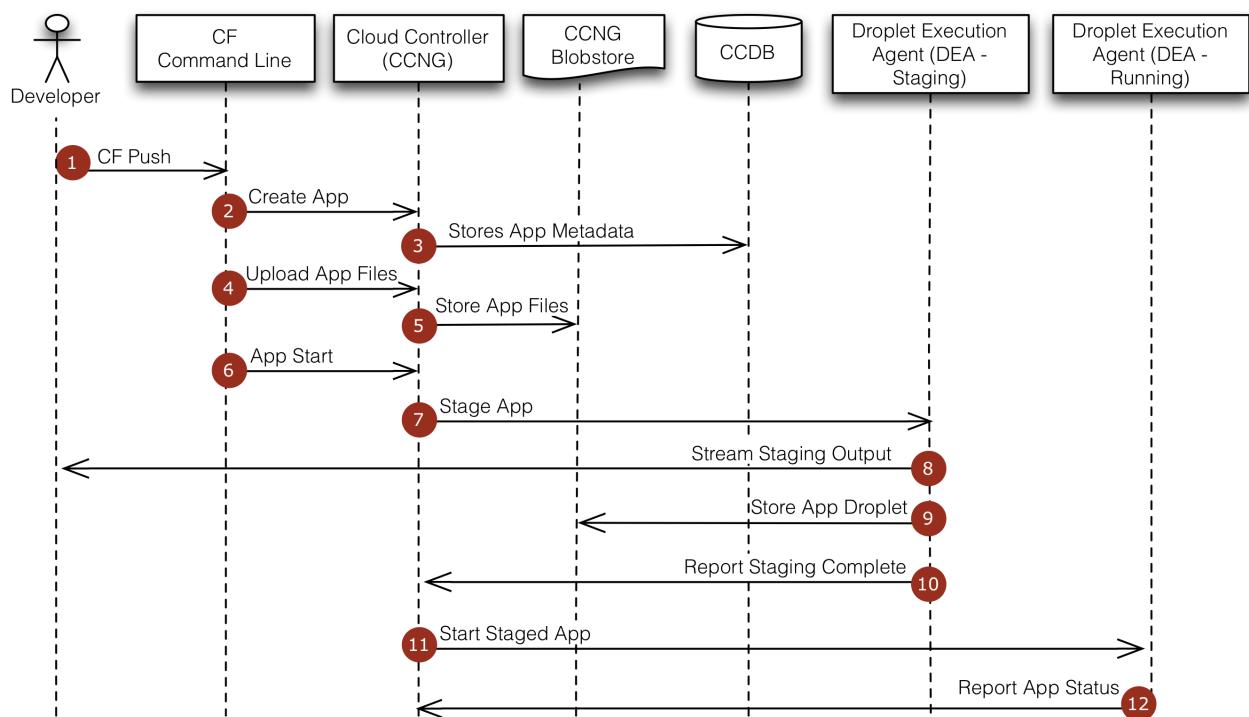
# start warden
$ cd /warden/warden
$ bundle install
$ rvmsudo bundle exec rake warden:start[config/test_vm.yml] > /tmp/warden.log &

# start the DEA's dependencies
$ cd /vagrant
$ bundle install
$ git submodule update --init
$ foreman start > /tmp/foreman.log &

# run the dea tests
$ bundle exec rspec
```

Staging Apps with DEAs

This section describes how the DEA architecture stages applications.



1. At the command line, the developer enters the directory containing her application and uses the Cloud Foundry Command Line Interface (cf CLI) tool to issue a push command.
2. The cf CLI tool tells the Cloud Controller to create a record for the application.
3. The Cloud Controller stores the application metadata (e.g. the app name, number of instances the user specified, and the buildpack).
4. The cf CLI tool uploads the application files.
5. The Cloud Controller stores the raw application files in the blobstore.
6. The cf CLI tool issues an app start command.

7. Because the app has not already been staged, the Cloud Controller chooses a DEA instance from the DEA pool to stage the application. The staging DEA uses the instructions in the buildpack to stage the application.
8. The staging DEA streams the output of the staging process so the developer can troubleshoot application staging problems.
9. The staging DEA packages the resulting staged application into a tarball called a “droplet” and stores it in the blobstore. The results are cached and used next time the application is staged.
10. The staging DEA reports to the Cloud Controller that staging is complete.
11. The Cloud Controller chooses one or more DEAs from the pool to run the staged application.
12. The running DEAs report the status of the application to the Cloud Controller.

A DEA publishes the following staging messages on NATS:

- `staging.advertise`: Stagers broadcast their capacity/capability.
- `staging.locate`: Stagers respond to any message on this subject with a `staging.advertise` message. CC uses this to bootstrap.
- `staging.UUID.start`: Stagers respond to requests on this subject to stage applications.
- `staging`: Stagers, in a queue group, respond to requests to stage an application. Note: This is an old protocol.

For more information about how the Cloud Controller uses an algorithm to schedule apps on DEAs, see the [DEA Placement Algorithm](#) topic.

Logs

[Steno](#) handles the DEA logging. DEA logs to STDOUT by default. You can configure the DEA to log to a file, a syslog server, or both. The logging level specifies the verbosity of the logs, for example `warn`, `info`, or `debug`.

Component: DEA Placement Algorithm

Page last updated:

This topic describes how the Cloud Controller uses an algorithm to schedule apps in Cloud Foundry's older Droplet Execution Agent (DEA) architecture. Newer versions of Cloud Foundry use the [Diego Auction](#).

DEA Placement Algorithm

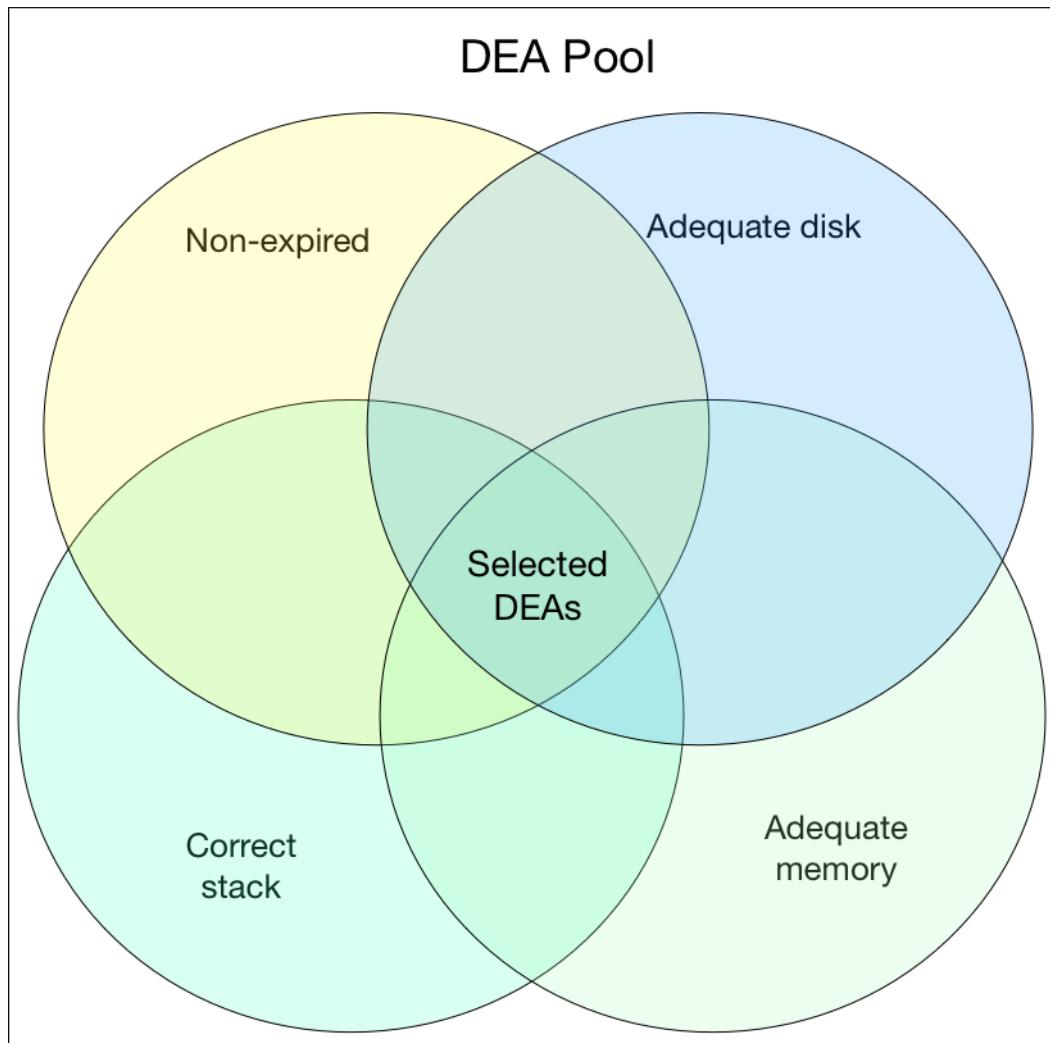
 **Note:** Pivotal Cloud Foundry (PCF) versions 1.6 and above no longer use the DEA Placement Algorithm. The [Diego Auction](#) replaces its function. This section is published here to help people who are running older versions of PCF.

Whenever Cloud Foundry needs to spin up a new instance of an app, the Cloud Controller is responsible for [selecting a droplet execution agent \(DEA\) to run it](#).

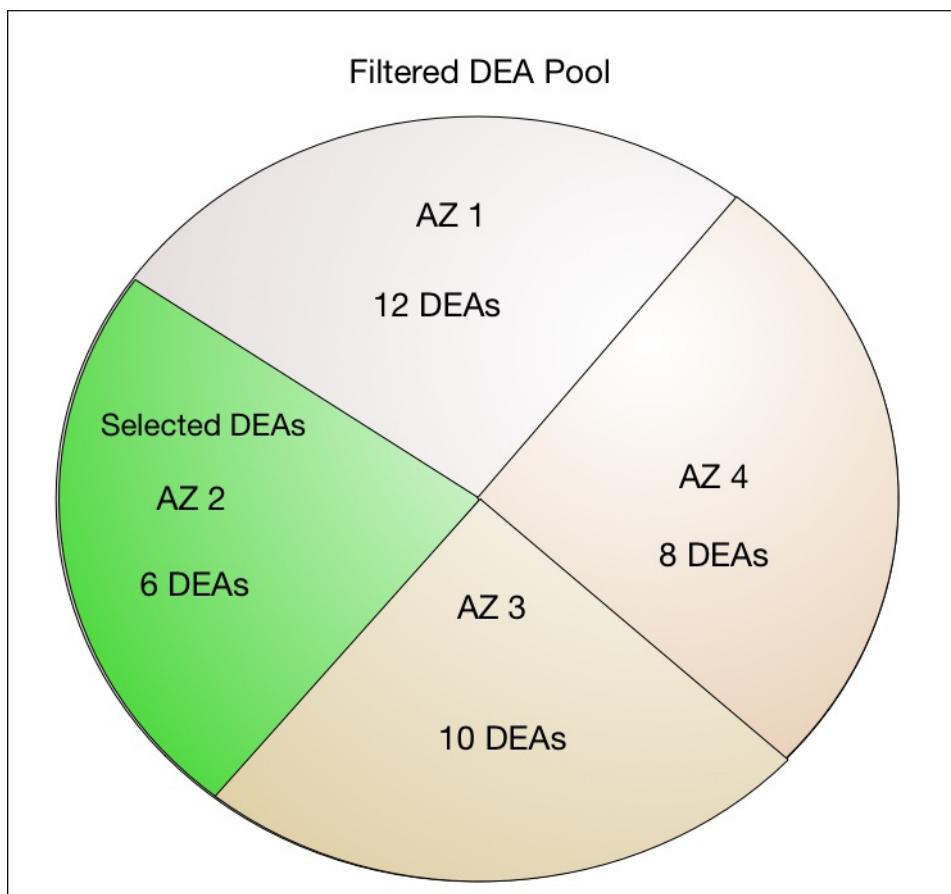
DEAs broadcast their availability to the Cloud Controller through a NATS message called an `advertisement`, which contains a `stats` hash with information about their available memory, available disk, the stack which the DEA runs, and an expiration time.

The Cloud Controller collects these advertisements in a construct called a pool. When the Cloud Controller needs to find a DEA to run an app, it runs through the following steps, using criteria (minimum thresholds for disk, memory, etc.) specific to the app that the chosen DEA will run:

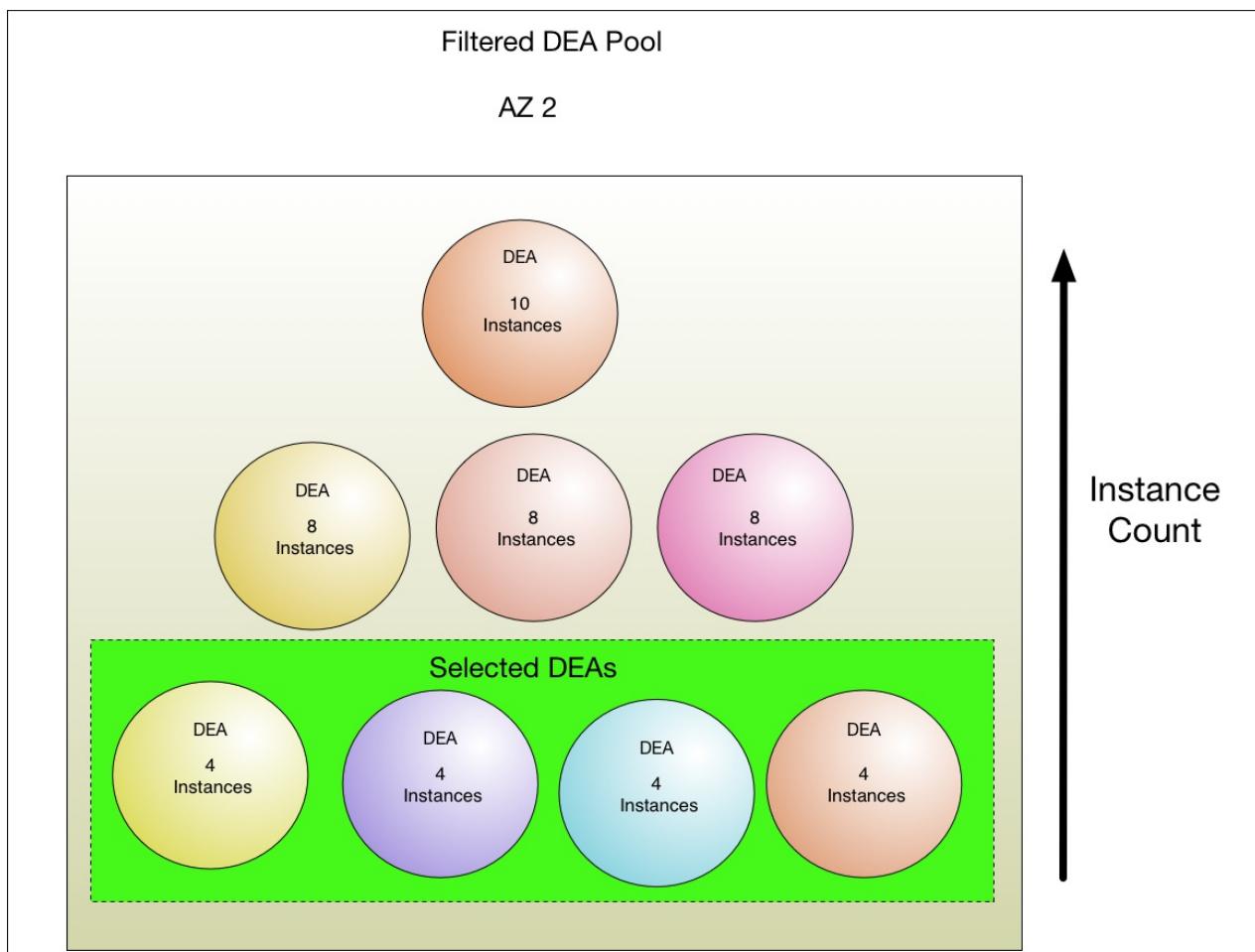
1. It removes the expired DEA advertisements from the pool.
2. It filters the remaining advertisements to include only those:
 - With adequate disk
 - With adequate memory
 - Running the required stack (Linux or Windows)



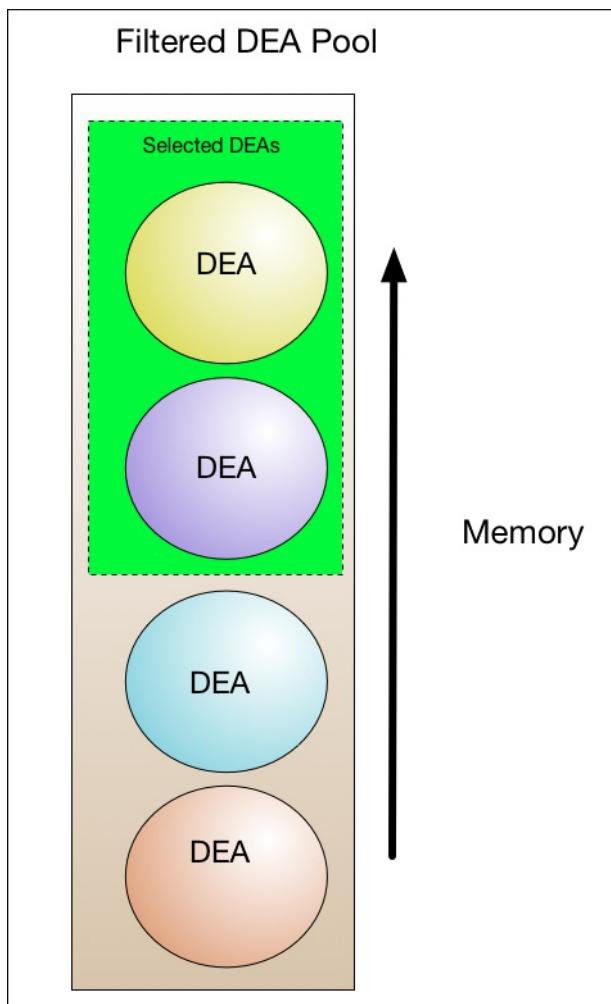
3. It then narrows its search to DEAs running in the availability zone with the fewest running instances (according to the information provided by the advertisements in the pool).



- It then narrows its search to the DEAs with the fewest running instances.



5. It then narrows its search to the top half of the DEAs, sorted by memory.



6. It then randomly selects one of the remaining DEAs.

Considerations

It is important to note that the Cloud Controller uses this algorithm to balance new app instances between DEAs when the new app instances are created, but do not balance already-running apps.

For example, suppose a set of apps are running on DEAs in two AZs, and one AZ temporarily goes down. While the second AZ is down, all instances will be placed on the remaining AZ. After the second AZ comes back online, new instances will be allocated to DEAs there, since the algorithm favors DEAs in the zone with the fewest running instances. However, instances running on the first AZ will not be moved to the other AZ, so the imbalance will persist.

An imbalance may also result from a deploy where DEAs have had a change to their source code or stemcell.

It is possible to rebalance the DEAs between AZs in two ways:

- Restarting the app, which may result in a brief down-time.
- Terminating and restarting half of the instances one by one.

Component: Warden

Page last updated:

 **Note:** This topic refers to versions of Cloud Foundry that use the older Droplet Execution Agent (DEA) architecture instead of the newer Diego architecture. In Diego, [Garden](#) replaces Warden.

Warden manages isolated, ephemeral, and resource-controlled environments.

The primary goal of Warden is to provide a simple API for managing isolated environments. These isolated environments, or *containers*, can be limited in terms of CPU usage, memory usage, disk usage, and network access. The only currently supported OS is Linux.

 **Note:** Pivotal Cloud Foundry versions 1.6 and above no longer use Warden. The platform-independent Garden subsystem replaces its function. This topic is intended for users who are running older versions of PCF.

Components

The [Warden](#) repository in GitHub contains the following components:

- `warden`: Server
- `warden-protocol`: Protocol definition, used by both the server and clients
- `warden-client`: Client (Ruby)
- `em-warden-client`: Client (Ruby's EventMachine)

Getting Started

This guide assumes Ruby 1.9 and Bundler are already available. Ensure that Ruby 1.9 has GNU readline library support through the `libreadline-dev` package and `zlib` support through the `zlib1g-dev` package.

Install Kernel

If you are running Ubuntu 10.04 (Lucid), make sure the backported Natty kernel is installed. After installing, reboot the system before continuing.

```
$ sudo apt-get install -y linux-image-generic-lts-backport-natty
```

Install Dependencies

```
$ sudo apt-get install -y build-essential debootstrap
```

Set up Warden

Run the `setup` routine. The `setup` routine compiles the C code bundled with Warden and sets up the base file system for Linux containers.

```
$ sudo bundle exec rake setup[config/linux.yml]
```

 **Note:** If `sudo` complains that `bundle` cannot be found, use `sudo env PATH=$PATH` to pass your current `PATH` to the `sudo` environment.

The `setup` routine sets up the file system for the containers at the directory path specified under the key `server -> container_rootsfs_path` in the config file `config/linux.yml`.

Run Warden

```
$ sudo bundle exec rake warden:start[config/linux.yml]
```

Interact with Warden

```
$ bundle exec bin/warden
```

Linux Implementation

Each application deployed to Cloud Foundry runs within its own self-contained environment, a Linux Warden container. Cloud Foundry operators can configure whether contained applications can or cannot directly interact with other applications or other Cloud Foundry system components.

Applications are typically allowed to invoke other applications in Cloud Foundry by leaving the system and re-entering through the Load Balancer positioned in front of the Cloud Foundry routers. The Warden containers isolate processes, memory, and the file system. Each Warden container also provides a dedicated virtual network interface that establishes IP filtering rules for the app, which are derived from network traffic rules. This restrictive operating environment is designed for security and stability.

Isolation is achieved by namespacing kernel resources that would otherwise be shared. The intended level of isolation is set such that multiple containers present on the same host are not aware of each other's presence. This means that these containers are given their own Process ID (PID), namespace, network namespace, and mount namespace.

Resource control is managed through the use of Linux Control Groups ([cgroups](#)). Every container is placed in its own control group, where it is configured to use a fair share of CPU compared to other containers and their relative CPU share, and the maximum amount of memory it may use.

CPU

Each container is placed in its own cgroup. Cgroups make each container use a fair share of CPU relative to the other containers. This prevents oversubscription on the host VM where one or more containers hog the CPU and leave no computing resources to the others.

The way cgroups allocate CPU time is based on shares. CPU shares do not work as direct percentages of total CPU usage. Instead, a share is relative in a given time window to the shares held by the other containers. The total amount of CPU that can be overall divided among the cgroups is what is left by other processes that may run in the host VM.

Generally, cgroups offers two possibilities for limiting the CPU usage: *CPU affinity* and *CPU bandwidth*, the latter of which is used in Cloud Foundry.

- CPU affinity: It consists of binding a cgroup to certain CPU cores. The actual amount of CPU cycles that can be consumed by the cgroup is thus limited to what is available on the bound CPU cores.
- CPU bandwidth: Sets the weight of a cgroup with the process scheduler. The process scheduler divides the available CPU cycles among cgroups depending on the shares held by each cgroup, relative to the shares held by the others.
For example, consider two cgroups, one holding two shares and one holding four. Assuming the process scheduler gets to administer 60 CPU cycles, the first cgroup with two shares will get one third of those available CPU cycles, as it holds one third of the overall shares. Similarly, the second cgroup will get 40 cycles, as it holds two thirds of the collective shares.

The calculation of the CPU usage based on the percentage of the total CPU power available is quite sophisticated and is performed regularly as the CPU demands of the various containers fluctuates. Specifically, the percentage of CPU cycles a cgroup gets can be calculated by dividing the `cpu.shares` it holds by the sum of the `cpu.shares` of all the cgroups that are currently doing CPU activity:

```
process_cpu_share_percent = cpu.shares / sum_cpu_shares *  
100
```

In Cloud Foundry, cgroup shares range from 2/10 (for DEA/Diego, respectively) to 1024, with 1024 being the default. The actual amount of shares a cgroup gets can be read from the `cpu.shares` of the c-group configurations pseudo-file-system located in the host VM under the `/tmp/warden/cgroups` folder.

The actual amount of shares given to a cgroup depends on the amount of memory the application declares to need in the manifest. Both Diego and DEA apps scale the number of allocated shares linearly with the amount of memory, with an app instance requesting 8G of memory getting the upper limit of 1024 shares.

```
process_cpu.shares = max( application_memory / 8,  
1024)
```

The next example helps to illustrate this better. Consider three processes, P1, P2 and P3, which are assigned `cpu.shares` of 5, 20 and 30, respectively.

P1 is active, while P2 and P3 require no CPU. Hence, P1 may use the whole CPU. When P2 joins in and is doing some actual work (e.g. a request comes in), the CPU share between P1 and P2 will be calculated as follows:

- P1 -> $5 / (5+20) = 0.2 = 20\%$
- P2 -> $20 / (5+20) = 0.8 = 80\%$
- P3 (idle)

At some point process P3 joins in as well. Then the distribution will be recalculated again:

- P1 -> $5 / (5+20+30) = 0.0909 = \sim 9\%$
- P2 -> $20 / (5+20+30) = 0.363 = \sim 36\%$
- P3 -> $30 / (5+20+30) = 0.545 = \sim 55\%$

Should P1 become idle, the following recalculation between P2 and P3 takes place:

- P1 (idle)
- P2 -> $20 / (20+30) = 0.4 = 40\%$
- P3 -> $30 / (20+30) = 0.6 = 60\%$

If P3 finishes or becomes idle then P2 can consume all the CPU as another recalculation will be performed.

 **Summary:** The cgroup shares are the minimum guaranteed CPU share that the process can get.

Networking

The DEA for each Warden container assigns a dedicated virtual network interface that is one side of a virtual ethernet pair created on the host. The other side of the virtual ethernet pair is only visible on the host from the root namespace. The pair is configured to use IPs in a small and static subnet. Traffic to and from the container is forwarded using NAT. Operators can configure global and container-specific network traffic rules that become Linux iptable rules to filter and log outbound network traffic.

Filesystem

Every container receives a private root filesystem. For Linux containers, this filesystem is created by stacking a read-only base filesystem and a container-specific read-write filesystem, commonly known as an overlay filesystem.

The read-only filesystem contains the minimal set of Linux packages and Warden-specific modifications common to all containers. Containers can share the same read-only base filesystem because all writes are applied to the read-write filesystem.

The read-write filesystem is unique to each container and is created by formatting a large sparse file of a fixed size. This fixed size prevents the read-write filesystem from overflowing into unallocated space.

Difference with LXC

The *Linux Containers* or *LXC* project has goals that are similar to those of Warden: isolation and resource control. They both use the same Linux kernel primitives to achieve their goals. Early versions of Warden used LXC.

The major difference between the two projects is that LXC is explicitly tied to Linux, while you can implement Warden backends for any operating system that implements some way of isolating environments. It is a daemon that manages containers and can be controlled through a simple API, rather than a set of tools that are individually executed.

While the Linux backend for Warden was initially implemented with LXC, the current version no longer depends on it. Because Warden relies on a very small subset of the functionality that LXC offers, we decided to create a tool that only implements the functionality we need in under 1k LOC of C code. This tool executes preconfigured hooks at different stages of the container start process, such that required resources can be set up without worrying about concurrency issues. These hooks make the start process more transparent, allowing for easier debugging when

parts of this process are not working as expected.

Container Lifecycle

Warden manages the entire lifecycle of containers. The API allows users to create, configure, use, and destroy containers. Additionally, Warden can automatically clean up unused containers when needed.

Create

Every container is identified by its *handle*, which Warden returns upon creating it. The handle is a hexadecimal representation of the IP address that is allocated for the container. Regardless of whether the backend providing the container functionality supports networking, Warden allocates an IP address to identify a container.

Once a container is created and its handle is returned to the caller, it is immediately ready for use. All resources will be allocated, the necessary processes will be started and all firewalling tables will have been updated.

If Warden is configured to clean up containers after activity, it will use the number of connections that have referenced the container as a metric to determine inactivity. If the number of connections referencing the container drops to zero, the container will automatically be destroyed after a preconfigured interval. If in the mean time the container is referenced again, this timer is cancelled.

Use

The container can be used by running arbitrary scripts, copying files in and out, modifying firewall rules and modifying resource limits. Refer to the [Interface](#) section for a complete list of operations.

Destroy

When a container is destroyed, either per user request, or automatically after being idle, Warden first kills all unprivileged processes running inside the container. These processes first receive a `TERM` signal, followed several seconds later by a `KILL` signal if they have not yet exited. When these processes have terminated, Warden sends a `KILL` to the root of the container process tree. Once all resources the container used have been released, its files are removed and it is considered destroyed.

Networking

Interface

Warden uses a line-based JSON protocol to communicate with its clients, which it does over a Unix socket located at `/tmp/warden.sock` by default. Each command invocation is formatted as a JSON array, where the first element is the command name and subsequent elements can be any JSON object. Warden responds to the following commands:

create [CONFIG]

Creates a new container.

Returns the container handle, which is used to identify the container.

The optional `[CONFIG]` parameter is a hash that specifies configuration options used during container creation. The supported options are:

bind_mounts

If supplied, this specifies a set of paths to be bind mounted inside the container. The value must be an array. The elements in this array specify the bind mounts to execute, and are executed in order. Every element must be of the form:

```
[
  # Path in the host filesystem
  "/host/path",
  # Path in the container
  "/path/in/container",
  # Optional hash with options. The 'mode' key specifies whether the bind
  # mount should be remounted as 'ro' (read-only) or 'rw' (read-write).
  {
    "mode" => "ro|rw"
  }
]
```

grace_time

If specified, this setting overrides the default time period during which no client references a container before the container is destroyed. The value can either be the number of seconds as floating point number or integer, or the `null` value to completely disable the grace time.

disk_size_mb

If specified, this setting overrides the default size of the container scratch filesystem. The value is expected to be an integer number.

spawn HANDLE SCRIPT [OPTS]

Run the script `SCRIPT` in the container identified by `HANDLE`.

Returns a job identifier that can be used to reap the job exit status at some point in the future. Also, the connection that issued the command may disconnect and reconnect later, but still be able to reap the job.

The optional `[OPTS]` parameter is a hash that specifies options modifying the command being run. The supported options are:

privileged

If true, this specifies that the script should be run as root.

link HANDLE JOB_ID

Reap the script identified by `JOB_ID`, running in the container identified by `HANDLE`.

Returns a three-element tuple containing the integer exit status, a string containing its `STDOUT` and a string containing its `STDERR`. These elements may be `null` when they cannot be determined (e.g. the script could not be executed, was killed, etc.).

stream HANDLE JOB_ID

Stream `STDOUT` and `STDERR` of scripts identified by `JOB_ID`, running in the container identified by `HANDLE`.

Returns a two-element tuple containing the type of stream: `STDOUT` or `STDERR` as the first element, and a chunk of the stream as the second element. Returns an empty tuple when no more data is available in the stream.

limit HANDLE (mem) [VALUE]

Set or get resource limits for the container identified by `HANDLE`.

The following resources can be limited:

- The memory limit is specified in number of bytes. It is enforced using the control group associated with the container. When a container exceeds this limit, the kernel kills one or more of its processes. Additionally, the Warden is notified that an OOM happened. The Warden

subsequently tears down the container.

net HANDLE in

Forward a port on the external interface of the host to the container identified by `HANDLE`.

Returns the port number that is mapped to the container. This port number is the same on the inside of the container.

net HANDLE out ADDRESS[/MASK][:PORT]

Allow traffic from the container identified by `HANDLE` to the network address specified by `ADDRESS`. Additionally, the address may be masked to allow a network of addresses, and a port to only allow traffic to a specific port.

Returns `ok`.

copy HANDLE in SRC_PATH DST_PATH

Copy the contents at `SRC_PATH` on the host to `DST_PATH` in the container identified by `HANDLE`.

Returns `ok`.

File permissions and symbolic links are preserved, while hardlinks are materialized. If `SRC_PATH` contains a trailing `/`, only the contents of the directory are copied. Otherwise, the outermost directory, along with its contents, is copied. The unprivileged user will own the files in the container.

copy HANDLE out SRC_PATH DST_PATH [OWNER]

Copy the contents at `SRC_PATH` in the container identified by `HANDLE` to `DST_PATH` on the host.

Returns `ok`.

Its semantics are identical to `copy HANDLE in` except with respect to file ownership. By default, root owns the files on the host. If you supply the `OWNER` argument (in the form of `USER:GROUP`), files on the host will be chowned to this user/group after the copy has completed.

stop HANDLE

Stop processes running inside the container identified by `HANDLE`.

Returns `ok`.

Because this takes down all processes, unfinished scripts will likely terminate without an exit status being available.

destroy HANDLE

Stop processes and destroy all resources associated with the container identified `HANDLE`.

Returns `ok`.

Because everything related to the container is destroyed, artifacts from running an earlier script should be copied out before calling `destroy`.

Configuration

You can configure Warden by passing a configuration file when it starts. Refer to `config/linux.yml` in the repository for an example configuration.

System Prerequisites

Warden runs on Ubuntu 10.04 and higher.

A backported kernel needs to be installed on 10.04. This kernel is available as `linux-image-server-lts-backport-natty` (substitute `generic` for `server` if you are running Warden on a desktop variant of Ubuntu 10.04).

Other dependencies are:

- `build-essential` (for compiling the Warden C bits)
- `debootstrap` (for bootstrapping the base filesystem of the container)
- `quota` (for managing file system quotas)

Run `rake setup` for further Warden bootstrapping.

Hacking

The included tests create and destroy real containers, so they require system prerequisites to be in place. They need to be run as root if the backend to be tested requires it.

See `root/<backend>/README.md` for backend-specific information.

Diego Architecture

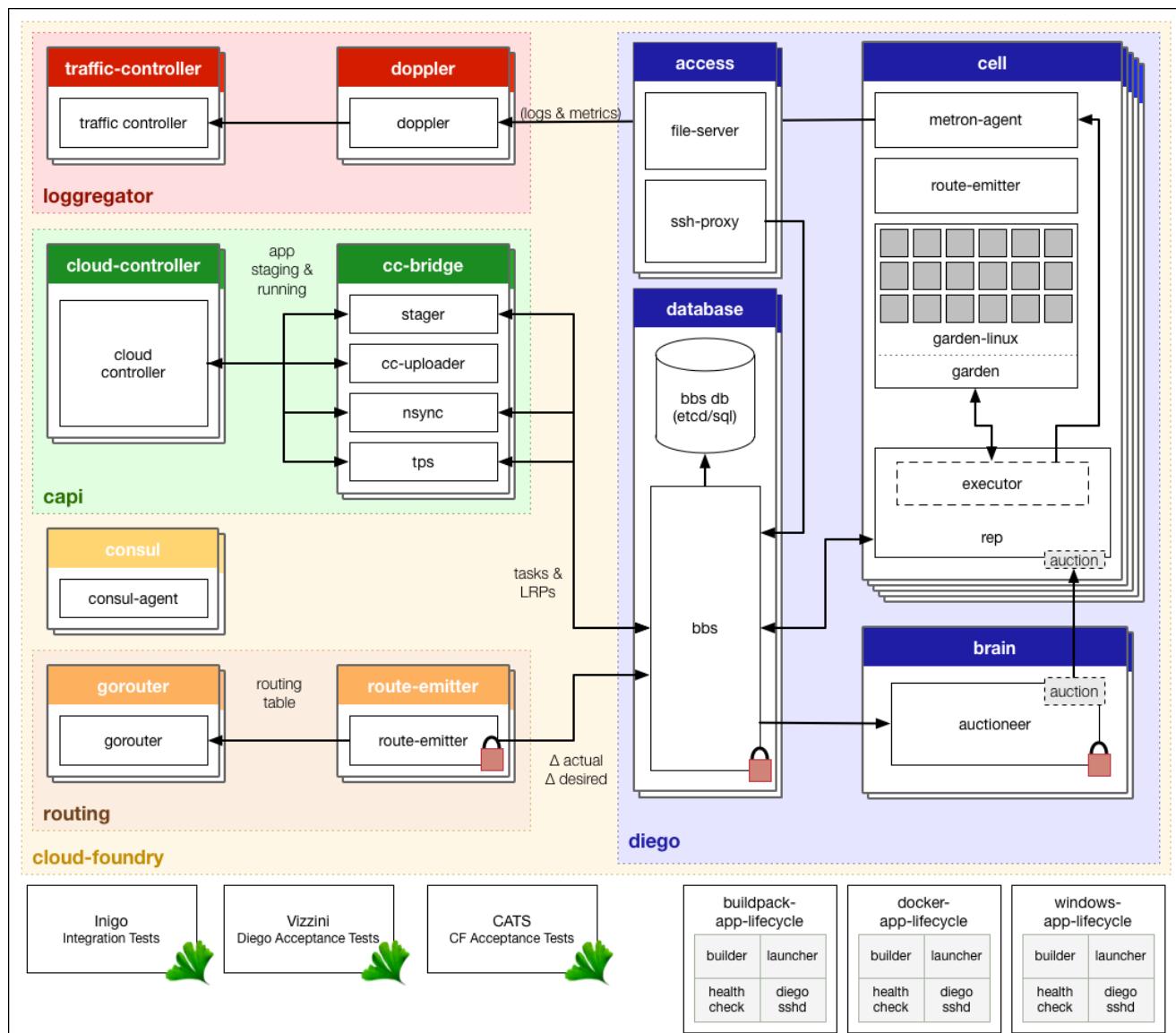
Page last updated:

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.

Diego Architecture

Cloud Foundry has used two architectures for managing application containers: [Droplet Execution Agents](#) (DEA) and Diego. With the DEA architecture, the [Cloud Controller](#) schedules and manages applications on the DEA nodes. In the newer Diego architecture, Diego components replace the DEAs and the [Health Manager \(HM9000\)](#), and 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 at the [Diego Design Notes repo](#).

1. The Cloud Controller passes requests to stage and run applications to the [Cloud Controller Bridge](#) (CC-Bridge).
2. The CC-Bridge translates staging and running requests into [Tasks and Long Running Processes](#) (LRPs), then submits these to the [Bulletin Board System](#) (BBS) through an API over HTTP.

3. The BBS submits the Tasks and LRPCs to the [Auctioneer](#), part of the [Diego Brain](#).
4. The Auctioneer distributes these Tasks and LRPCs 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 LRPCs, running LRPC 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 Core Components

Components in the Diego core run and monitor Tasks and LRPCs. The core consists of the following major areas:

- [Brain](#)
- [Cells](#)
- [Database VMs](#)
- [Access VMs](#)
- [Consul](#)

Diego Brain

Diego Brain components distribute Tasks and LRPCs 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 Auctioneer.

Auctioneer

- Uses the [auction package](#) to run Diego Auctions for Tasks and LRPCs
- 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 repo](#) on GitHub for more information.

Diego Cell Components

Diego Cell components manage and maintain Tasks and LRPCs.

Rep

- Represents a Cell in Diego Auctions for Tasks and LRPCs
- Mediates all communication between the Cell and the BBS
- Ensures synchronization between the set of Tasks and LRPCs in the BBS with the containers present on the Cell
- Maintains the presence of the Cell in the BBS
- Runs Tasks and LRPCs by asking the in-process Executor to create a container and [RunAction](#) recipes

Refer to the [Rep repo](#) 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 repo](#) 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 repo](#) on GitHub for more information.

Database VMs

Diego Bulletin Board System

- Maintains a real-time representation of the state of the Diego cluster, including all desired LRPCs, 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](#), [CC-Bridge](#), and [Route Emitter](#).
- Ensures consistency and fault tolerance for Tasks and LRPCs 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 repo](#) on GitHub for more information.

MySQL

- Provides a consistent key-value data store to Diego

Access VMs

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 repo](#) 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 repo](#) for more information.

Consul

- Provides dynamic service registration and load balancing through DNS resolution
- Provides a consistent key-value store for maintenance of distributed locks and component presence

Refer to the [Consul repo](#) on GitHub for more information.

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 repo](#) on GitHub for more information.

Cloud Controller Bridge Components

The Cloud Controller Bridge (CC-Bridge) components translate app-specific requests from the Cloud Controller to the BBS. These components include the following:

Stager

- Translates staging requests from the Cloud Controller into generic Tasks and LRPCs
- Sends a response to the Cloud Controller when a Task completes

Refer to the [Stager repo](#) 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 repo](#) on GitHub for more information.

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 repo](#) on GitHub for more information.

TPS

- Provides the Cloud Controller with information about currently running LRPCs to respond `tc_cf_apps` and `cf_app APP_NAME` requests
- Monitors `ActualLRP` activity for crashes and reports them to the Cloud Controller

Refer to the [TPS repo](#) 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 [CC-Bridge](#) runs the Builder as a Task on every staging request. The Builder 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 CC-Bridge sets the Launcher as the Action on the [DesiredLRP](#) for the application. The Launcher 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 [CC-Bridge](#) sets the Healthcheck 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.

Other Components

Route-Emitter

- Acts as either one global route-emitter or local route-emitters on each Diego cell
- Monitors [DesiredLRP](#) and [ActualLRP](#) states, emitting route registration and unregistration messages to the Cloud Foundry [router](#) when it detects changes
- Periodically emits the entire routing table to the Cloud Foundry router

Refer to the [Route-Emitter repo](#) on GitHub for more information.

Understanding Application SSH

Page last updated:

This document describes details about the Elastic Runtime SSH components for access to deployed application instances. Elastic Runtime supports native SSH access to applications and load balancing of SSH sessions with the load balancer for your Elastic Runtime deployment.

The [SSH Overview](#) document describes procedural and configuration information about application SSH access.

SSH Components

The Elastic Runtime 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 Elastic Runtime. 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 LRPCs that want to use it can include a download action to acquire the binary and a run action to start it. Elastic Runtime 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 Elastic Runtime. 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 the Diego Auction Allocates Jobs

Page last updated:

The [Diego](#) Auction balances application processes, also called jobs, over the virtual machines (VMs) in a Cloud Foundry installation. 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 application availability and resilience. This topic explains how the Diego Auction works at a conceptual level.

Refer to the [Auction repo](#) 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 **currentbatch** 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:

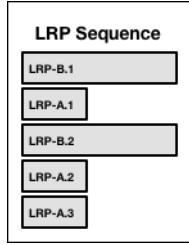
1. Keep at least one instance of each LRP running.
2. Run all of the Tasks in the current batch.
3. Distribute as much of the total desired LRP load as possible over the remaining available VMs, by spreading multiple LRP instances broadly across VMs and their Availability Zones.

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:

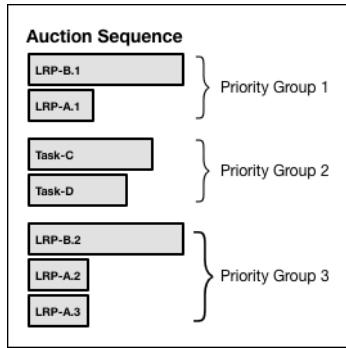
Process	Desired Instances	Load	Jobs
LRP-A	3	2	
LRP-B	2	5	

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:



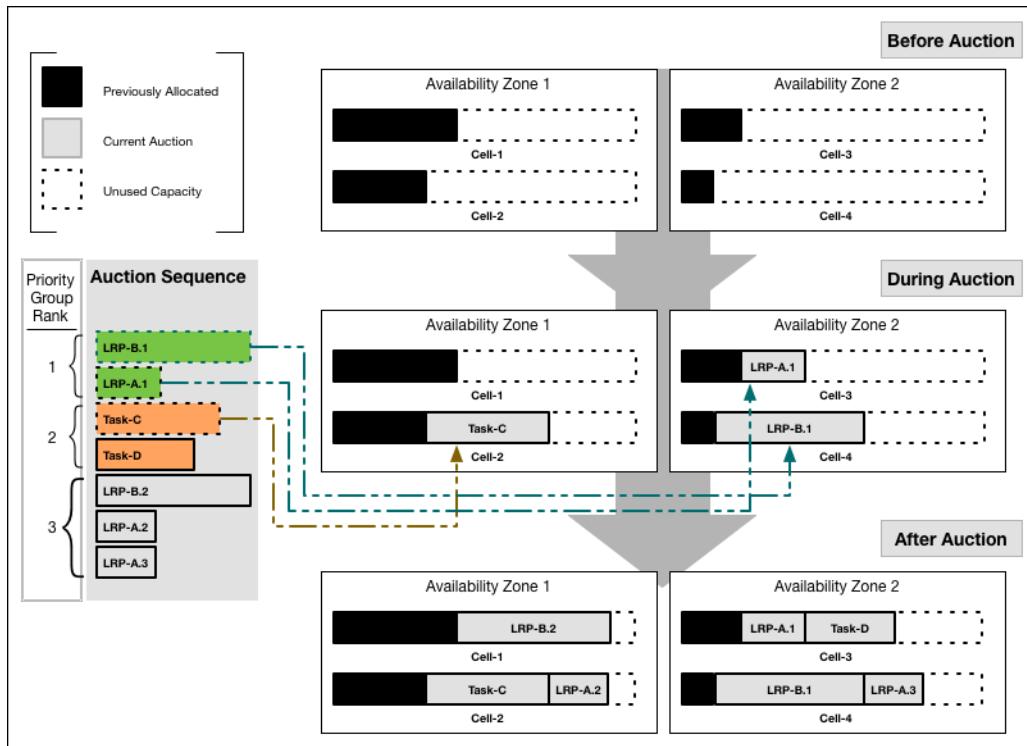
Auctioning the Batch to the Cells

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.

Operator's Guide

This guide covers networking and user management for [Pivotal Cloud Foundry](#) (PCF) operators.

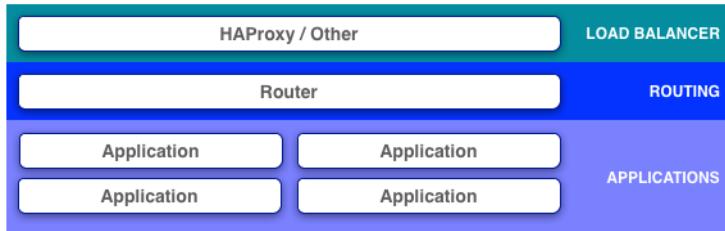
Table of Contents

- [Understanding the Elastic Runtime Network Architecture](#)
- [Identifying the API Endpoint for your Elastic Runtime Instance](#)
- [Creating New Elastic Runtime User Accounts](#)
- [Configuring SSL/TLS Termination at HAProxy](#)
- [Configuring Proxy Settings for All Applications](#)
- [Configuring Application Security Groups for Email Notifications](#)
- [Configuring SSH Access for PCF](#)
- [Identifying Elastic Runtime Jobs Using vCenter](#)
- [Configuring System Logging in Elastic Runtime](#)
- [Configuring UAA Password Policy](#)
- [Configuring Authentication and Enterprise SSO for Elastic Runtime](#)
- [Adding Existing SAML or LDAP Users to a Pivotal Cloud Foundry Deployment](#)
- [Switching Application Domains](#)
- [Scaling Elastic Runtime](#)
- [Scaling Down Your MySQL Cluster](#)
- [Using Docker Trusted Registries](#)
- [Custom Branding Apps Manager](#)
- [Monitoring App and Service Instance Usage](#)
- [Monitoring Instance Usage with Apps Manager](#)
- [Deploying Diego for Windows](#)
- [Operating Diego for Windows](#)
- [Monitoring a Pivotal Cloud Foundry Deployment](#)
- [Providing a Certificate for your SSL Termination Point](#)
- [Enabling NFS Volume Services](#)
- [Installing PCF Isolation Segment](#)

Understanding the Elastic Runtime Network Architecture

Page last updated:

The diagram below shows the key [Pivotal Cloud Foundry](#) (PCF) Elastic Runtime network components.



Load Balancer

Elastic Runtime includes an HAProxy load balancer for terminating SSL. If you do not want to serve SSL certificates for Elastic Runtime 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 Elastic Runtime 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 Elastic Runtime Instance

Page last updated:

The API endpoint for your Elastic Runtime 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 Elastic Runtime instance, then click **Tools** in the left navigation panel. The **Getting Started** section of the Tools page shows your API endpoint.

GETTING STARTED

```
$ cf help
$ cf login -a https://api.your_endpoint.com
API endpoint: https://api.your_endpoint.com
Username> your_username
Password> your_password
Org> your_org
Space> your_space
$ cf push your_app
```

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:

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.

CF 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.

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.

Terminate SSL/TLS at HAProxy

To use the HAProxy load balancer, you must create a wildcard A record in your DNS and configure some fields in the Elastic Runtime 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 Elastic Runtime 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
<code>*.cf</code>	A	<code>203.0.113.1</code>	<code>example.com</code>

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
```

In PCF, perform the following steps to configure SSL termination on HAProxy:

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the **Elastic Runtime** tile in the Installation Dashboard.
3. Click **Networking**.
4. For PCF deployments on OpenStack or vSphere, enter the HAProxy IP address that you used to set up a wildcard DNS record in the **HAProxy IPs** field. For more information, see the Elastic Runtime networking configuration topic for [OpenStack](#) or [vSphere](#). For PCF deployments on AWS or Azure, leave the HAProxy IP address blank.
5. Under **Select one of the point-of-entry-options** select the third option, **Forward SSL to HA Proxy**.
6. Enter your PEM-encoded certificate and your PEM-encoded private key in the fields under **SSL Termination Certificate and Private Key**. You can either upload your own certificate or generate a RSA certificate in Elastic Runtime. For options and instructions on creating a certificate for your wildcard domains, see [Creating a Wildcard Certificate for PCF Deployments](#)
7. If you want HAProxy to only allow HTTPS traffic, select **Disable HTTP traffic to HAProxy**.
8. If you want to use a specific set of SSL ciphers for HAProxy, configure **HAProxy SSL Ciphers**. Enter a colon-separated list of custom SSL ciphers to pass to HAProxy. Otherwise, leave this field blank.
9. If you expect requests larger than the default maximum of 16 Kbytes, enter a new value (in bytes) for **Request Max Buffer Size**. You may need to do this, for example, to support apps that embed large cookie or query string values in headers.
10. 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.

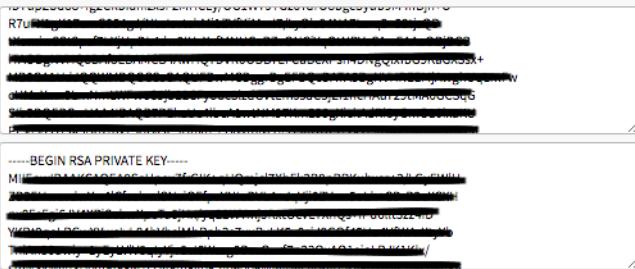
Select one of the following point-of-entry options:^{*}

Forward SSL to Elastic Runtime Router. Assumes an external load balancer is configured to forward encrypted traffic.

Forward unencrypted traffic to Elastic Runtime Router. Assumes an external load balancer is configured to forward unencrypted traffic.

Forward SSL to HA Proxy. Like first option - Assumes an external load balancer is configured to forward encrypted traffic.

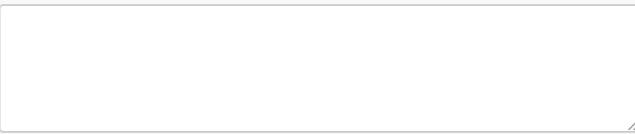
SSL Certificate and Private Key ^{*}



Generate RSA Certificate

Disable HTTP traffic to HAProxy

HAProxy SSL Ciphers



Request Max Buffer Size ^{*}

Disable SSL certificate verification for this environment

11. Complete the rest of the **Networking configuration** screen, and click **Save**.
12. For PCF deployments on Azure or AWS, configure the HAProxy job in the **Resource Config** page of Elastic Runtime tile. For more information, see the Elastic Runtime installation instructions for [Azure](#) or [AWS](#).

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 Elastic Runtime** 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.

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
<code>SMTP_SERVER</code>	587 (default)	tcp (default)	This service is used to send out email notifications
<code>LOAD_BALANCER_IP</code>	80, 443	tcp	This service will access the load balancer
<code>ASSIGNED_NETWORK</code>	3306	tcp	This service requires access to internal services. <code>ASSIGNED_NETWORK</code> 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 Elastic Runtime** 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 Elastic Runtime 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 Elastic Runtime 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.

Elastic Runtime Configuration

This section describes how to configure Elastic Runtime 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 Application. For details about SSH access permissions, see the [Application SSH Overview](#) topic.

To configure Elastic Runtime SSH access for application instances:

1. Open the **Pivotal Elastic Runtime** 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`.

Enable microservice frameworks, private Docker registries, and other services that support your applications at a container level.

- Enable Custom Buildpacks
- Allow SSH access to app containers
- Enable SSH when an app is created

Private Docker Insecure Registry Whitelist

10.10.10.10:8888,example.com:8888

Docker Images Disk-Cleanup Scheduling on Cell VMs*

- Never clean up Cell disk-space
- Routinely clean up Cell disk-space
- Clean up disk-space once threshold is reached

Threshold of Disk-Used (MB) (min: 1) *

Max Inflight Container Starts *

Save

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.

Identifying Elastic Runtime Jobs Using vCenter

Page last updated:

To effectively monitor, control, and manage the virtual machines making up your Elastic Runtime deployment, you may need to identify which VM corresponds to a particular job in Elastic Runtime. You can find the CID of a particular VM from [Pivotal Cloud Foundry](#) (PCF) Operations Manager by navigating to **Elastic Runtime > Status**.

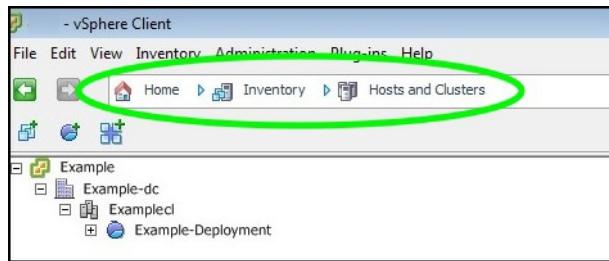
If you have deployed Elastic Runtime to VMware vSphere, you can also identify which Elastic Runtime job corresponds to which VM using the vCenter vSphere client.

 **Note:** The CID shown in Ops Manager is the name of the machine in vCenter.

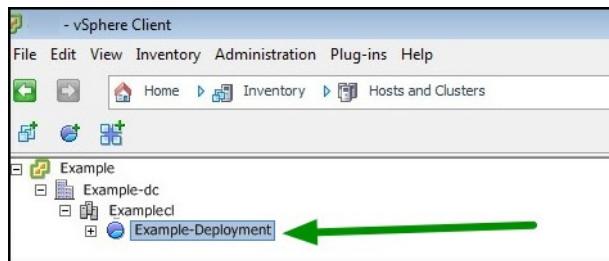
Identifying Elastic Runtime Jobs Using vCenter

1. Launch the vSphere client and log in to the vCenter Server system.

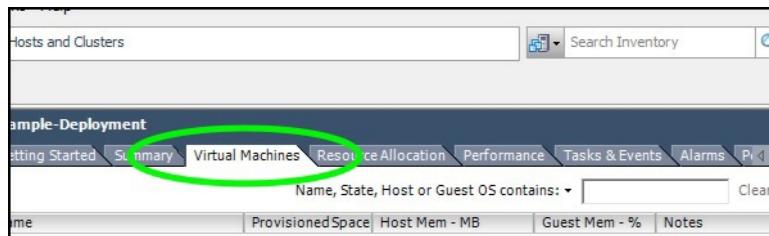
2. Select the **Inventory > Hosts and Clusters** view.



3. Select the Resource Pool containing your Elastic Runtime deployment.



4. Select the **Virtual Machines** tab.



5. Right-click the column label heading and check **job**.

Example-Deployment

Getting Started Summary Virtual Machines Resource Allocation Performance Tasks & Events Alarms Permissions Maps

Name, State, Host or Guest OS contains:

Name	Provisioned Space	Host Mem - MB	Guest
Host	9.09 GB	1040	6
Provisioned Space	11.09 GB	597	3
Used Space	151.09 GB	1036	45
Host CPU - MHz	10.09 GB	598	4
Host Mem - MB	19.09 GB	1023	86
Guest Mem - %	6.69 GB	0	0
Guest OS	9.09 GB	1045	1
VM Version	9.09 GB	470	1
Memory Size	9.09 GB	468	1
Reservation - MB	20.09 GB	3130	1
CPU Count	6.69 GB	0	0
NIC Count	9.09 GB	551	1
Uptime	6.69 GB	0	0
IP Address	6.69 GB	0	0
VMware Tools Running Status			
VMware Tools Version Status			
DNS Name			
EVC Mode			
UUID			
Notes			
Alarm Action			
vSphere HA Protection			
Needs Consolidation			
Name	Requested Start Time	Start Time	
comping	3/6/2014 12:21:36 PM	3/6/2014 12:21:36 PM	
deployment	3/6/2014 12:21:32 PM	3/6/2014 12:21:32 PM	
director			
index			
job			

Status

- Completed
- Completed

Notes

- alarm action
- vSphere HA protection
- needs consolidation
- name
- comping
- deployment
- director
- index
- job

6. The job column displays the Elastic Runtime job associated with each virtual machine.

Example-Deployment

Getting Started Summary Virtual Machines Resource Allocation Performance Tasks & Events Alarms Permissions Maps

Name, State, Host or Guest OS contains:

Name	job
vm-347a89c6-115f-433a-83d1-38cd2c31451b	ccdb
sc-d3520bac-11c2-460f-aa02-60fc11cba375	cloud_controller
vm-d537f816-fc51-450c-8ada-cb536a9e76f5	consoledb
vm-85d43147-7cd2-4cc5-acfc-47cce18cccd69	dea
vm-5c04a27a-ca70-4f67-a221-767fb76922a	ha_proxy
vm-e9b3e319-690f-44be-87fb-ce97f90c4b2	health_manager
vm-a8bd3d1-7e31-47f3-800f-3f81ff23ed81	loggregator
vm-ee8b5374-bbc9-436b-bfe5-1a7e2b5fee9a	loggregator_trafficcontroller
vm-5896908f-15c5-4989-8c90-36a36e0eb4fd	nats
vm-61fc0d19-01f5-4434-82e0-a11e15b38888	nfs_server
vm-e88725cd-36fe-4a79-b8ea-f35b2d2cd85b	riak-cs
vm-9ab66b39-4cae-443a-842c-3651c9bf228c	router
vm-2726dcda-7f92-440c-906e-3e5e544fe628	saml_login
vm-273df76f-7141-432d-9e53-9cb6549d870a	uaa
vm-307a34f4-ed7f-4c48-94d1-bb7a61a695bc	uaadb
vm-e6c2dfcd-ef2e-40d8-af0b-06e5f3e79fa9	
vm-0e9e9384-0057-4efc-9cea-234b5956c510	
vm-5e33e793-0118-44bb-948d-3ea6526b4b2f	

Configuring System Logging in Elastic Runtime

Page last updated:

This topic explains how to configure the Pivotal Cloud Foundry [Loggregator system](#) to scale its maximum throughput, and to forward logs to an external aggregator service.

Scaling Loggregator

Elastic Runtime system components and apps constantly generate log and metrics data. The [Metron](#) agent running on each component or application VM collects and sends this data out to [Doppler](#) components, which temporarily buffer the data before periodically forwarding it to the [Traffic Controller](#). The Traffic Controller then serves the aggregated data stream through the Firehose WebSocket endpoint.

When the log and metrics data input to a Doppler exceeds its buffer size for a given interval, data can be lost. You can take several actions to minimize this loss.

Increase buffer size

1. In the [Pivotal Cloud Foundry \(PCF\) Ops Manager Installation Dashboard](#), click the **Elastic Runtime** tile.
2. Select **System Logging**.
3. Increase the **Drain Buffer Size** to prevent loss of log data.
4. Click **Save**.
5. Click **Apply Changes**.

Configure system logging. Complete the External Syslog fields only if using an external syslogd server.

Enable Cloud Controller security event logging

External Syslog Aggregator Hostname

External Syslog Aggregator Port
 The typical syslog port is 514. Ensure syslogd is listening on external interfaces.

External Syslog Network Protocol

Syslog Drain Buffer Size (# of messages) *

Save

Add Additional Doppler Instances

1. In the [PCF Ops Manager Installation Dashboard](#), click the **Elastic Runtime** tile.
2. Select **Resource Config**.
3. Increase the number in the **Instances** column and the **Doppler Server** row.

<input checked="" type="checkbox"/> Resource Config	Doppler Server	Automatic: 1
	Loggregator Trafficcontroller	Automatic: 1

4. Click **Save**.

5. Click **Apply Changes**.

Add Additional Traffic Controller Instances

1. In the **PCF Ops Manager Installation Dashboard**, click the **Elastic Runtime** tile.

2. Select **Resource Config**.

3. Increase the number in the **Instances** column and the **Loggregator Trafficcontroller** row.

<input checked="" type="checkbox"/> Resource Config	Doppler Server	Automatic: 1
	Loggregator Trafficcontroller	Automatic: 1

4. Click **Save**.

5. Click **Apply Changes**.

Enabling System Log Forwarding

Elastic Runtime can forward log data to an external aggregator service instead of routing it to the Loggregator Firehose. System log forwarding for [Pivotal Cloud Foundry](#) (PCF) is managed through the **PCF Ops Manager Installation Dashboard**. Complete the steps below to enable syslog forwarding:

1. Click the **Elastic Runtime** tile.

2. Select **System Logging**.

3. If you want to include security events in your log stream, select the **Enable Cloud Controller security event logging** checkbox. The security log messages are in Common Event Format (CEF).

4. Enter the **Aggregator Hostname**, **Aggregator Port**, and **Network Protocol** for your third-party log management service.

5. (Optional) Increase the **Drain Buffer Size** to prevent loss of log data.

6. Click **Save**.

7. Click **Apply Changes**.

Configure system logging. Complete the External Syslog fields only if using an external syslogd server.

Enable Cloud Controller security event logging

External Syslog Aggregator Hostname

External Syslog Aggregator Port

The typical syslog port is 514. Ensure syslogd is listening on external interfaces.

External Syslog Network Protocol

Syslog Drain Buffer Size (# of messages) *

Save

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 Elastic Runtime 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 Elastic Runtime** tile.
3. Select **Authentication and Enterprise SSO** on the **Settings** tab.

Configure your user store access, which can be an internal user store (managed by Cloud Foundry's UAA) or an external user store (LDAP or SAML). You can also adjust the lifetimes of authentication tokens.

Configure your UAA user account store with either internal or external authentication mechanisms*

- Internal UAA (provided by Elastic Runtime; configure your password policy below)

Minimum Password Length *

0



Minimum number of characters for a valid password.

Minimum Uppercase Characters Required for Password *

0

Minimum Lowercase Characters Required for Password *

0

Minimum Numerical Digits Required for Password *

0

Minimum Special Characters Required for Password *

0

Number of Months Before Password Expires *

0

Maximum Password Entry Attempts Allowed *

5

4. Confirm that the **Internal UAA** option is selected.

Set Password Requirements

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.

Set Password Expiration and Entry Attempts

1. For **Number of Months Before Password Expires**, enter the number of months a password remains valid. Enter if you want passwords to never expire.
2. 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 Elastic Runtime

Page last updated:

This topic describes [Pivotal Cloud Foundry](#) (PCF) 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 Elastic Runtime 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 Deployment](#) topic 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 PCF to Use a SAML Identity Provider

To connect PCF Elastic Runtime with SAML, you must perform the following tasks:

1. [Configure PCF as a service provider for SAML](#)
2. [Configure SAML as an Identity Provider for PCF](#)

Configure PCF as a Service Provider for SAML

Follow the instructions below to configure PCF as a service provider for SAML.

1. From the **Installation Dashboard**, click the **Elastic Runtime** tile.
2. Select the **Domains** tab and record your system domain.

The screenshot shows the Pivotal Elastic Runtime Installation Dashboard. The top navigation bar has links for 'Installation Dashboard', 'Pivotal Elastic Runtime', 'Settings' (which is active and highlighted in blue), 'Status', 'Credentials', and 'Logs'. On the left, there's a sidebar with several configuration items: 'Assign AZs and Networks', 'Domains' (which is also highlighted in blue), 'Networking', 'Application Containers', 'Application Developer Controls', 'Application Security Groups', 'Authentication and Enterprise SSO', and 'Databases'. The main content area is titled 'Pivotal Elastic Runtime' and contains a section for 'Domains'. It says: 'Elastic Runtime hosts applications at subdomains under its apps domain and assigns system components to subdomains under its system domain. You need to configure a wildcard DNS for both the apps domain and system domain. The two domains can be the same, although this is not recommended.' Below this, there are two input fields: 'System Domain *' with the value 'sy' and 'Apps Domain *' with the value 'ap'. A note next to the Apps Domain field says: 'Default parent domain that pushed apps use for their hostnames. This domain also requires a wildcard DNS record. Use the Cloud Foundry command line interface (cf CLI) to add or delete subdomains assigned to individual apps.' At the bottom of the form is a blue 'Save' button.

3. Select **Authentication and Enterprise SSO**.
4. Select **SAML Identity Provider**.

 SAML Identity Provider

Provider Name *

Display Name *

Provider Metadata (if you would rather provide an SSO endpoint URL, skip to the next field)

Metadata XML content for the Identity Provider.

(OR) Provider Metadata URL

Name ID Format*

Email Address ▼

Email Domain(s)

First Name Attribute

Last Name Attribute

Email Attribute

External Groups Attribute

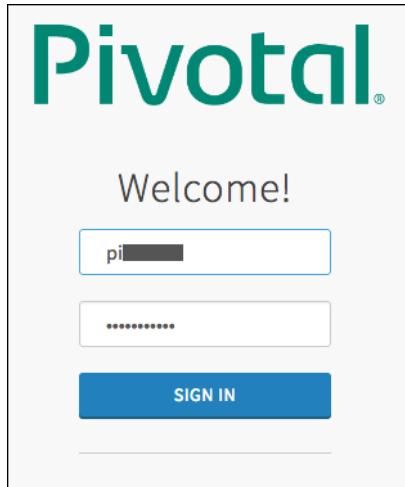
Sign Authentication Requests

Require Signed Assertions

Signature Algorithm*

SHA1 ▼

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`.

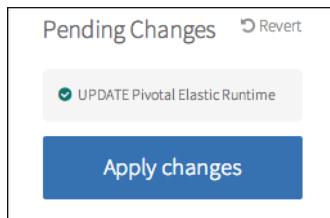


7. Retrieve the metadata from your Identity Provider and copy it into either the **Provider Metadata** or the **Provider Metadata URL** fields, depending on whether your Identity Provider exposes a Metadata URL. Refer to the [Configure SAML Identity Provider for PCF](#) section of this topic for more 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 PCF user roles.

8. Select the **Name ID Format** for your SAML Identity Provider. This translates to `username` on PCF Elastic Runtime. 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 PCF, 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 PCF 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. For **Signature Algorithm**, choose an algorithm from the dropdown menu to use for signed requests and assertions. The default value is `SHA1`.
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 PCF

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 PCF:

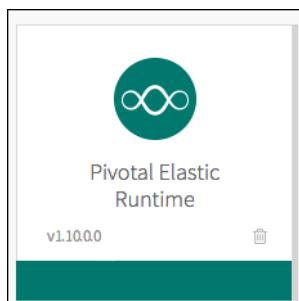
Solution Name	Integration Guide
CA Single Sign-On aka CA SiteMinder	PDF
Ping Federate	PDF
Active Directory Federation Services	PDF

 **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 PCF

To integrate the UAA with one or more LDAP servers, configure Elastic Runtime with your LDAP endpoint information as follows:

1. Log into the Operations Manager web interface.
2. On the Product Dashboard, select **Pivotal Elastic Runtime**.



3. In the left navigation menu, select **Authentication and Enterprise SSO**.

LDAP Server

Server URL *

LDAP Credentials *

Username

Password

User Search Base *

User Search Filter *

Group Search Base

Group Search Filter *

Server SSL Cert

Server SSL Cert AltName

First Name Attribute

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:

 **Note:** Pivotal recommends that you provide LDAP credentials that grant read-only permissions on the LDAP Search Base and the LDAP Group Search Base.

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:

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 returns all objects with a common name equal to .

In the LDAP search filter string that you use to configure Elastic Runtime, use instead of the username. For example, use 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.

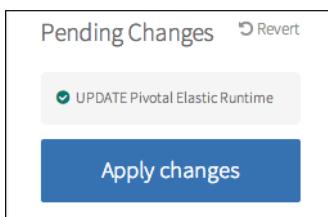
9. For **Group Search Base**, enter the location in the LDAP directory tree from which the LDAP Group search begins.

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 PCF.

 **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 PCF 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 Elastic Runtime** tile.
2. Select **Domains** from the menu to see the current **Apps Domain** for your Elastic Runtime deployment. In the following example it is `myapps.mydomain.com`.

Elastic Runtime hosts applications at subdomains under its apps domain and assigns system components to subdomains under its system domain. You need to configure a wildcard DNS for both the apps domain and system domain. The two domains can be the same, although this is not recommended.

System Domain *

Apps Domain *

 Default parent domain that pushed apps use for their hostnames. This domain also requires a wildcard DNS record. Use the Cloud Foundry command line interface (cf CLI) to add or delete subdomains assigned to individual apps.

Save

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 Elastic Runtime 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 Elastic Runtime 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:

APPLICATIONS	Learn More	
STATUS	APP	INSTANCES
100%	myapp myapp.newapps.mydomain...	1

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 Elastic Runtime tile to use the new domain, and apply changes. Apps that you push after your update finishes use this new domain.

Elastic Runtime hosts applications at subdomains under its apps domain and assigns system components to subdomains under its system domain. You need to configure a wildcard DNS for both the apps domain and system domain. The two domains can be the same, although this is not recommended.

System Domain *	<input type="text" value="mysystem.mydomain.com"/>	This domain is for system-level PCF components, such as Apps Manager, service brokers, etc. You must set up a wildcard DNS record for this domain that points to your entry point load balancer or HAProxy.
Apps Domain *	<input type="text" value="newapps.mydomain.com"/>	
<input type="button" value="Save"/>		

Scaling Elastic Runtime

Page last updated:

This topic discusses how to scale Elastic Runtime 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 Diego or PCF configuration highly available, see the [Zero Downtime Deployment and Scaling in CF](#) topic.

Steps for Scaling Elastic Runtime

1. Navigate to the Pivotal Cloud Foundry Operations Manager Installation Dashboard.
2. Click the Elastic Runtime tile in the Installation Dashboard.
3. Select **Resource Config**.

JOB	INSTANCES
Consul	Automatic: 1
NATS	Automatic: 1
etcd Server	Automatic: 1
etcd Proxy	Automatic: 1
File Storage	Automatic: 1
MySQL Proxy	Automatic: 1
MySQL Server	Automatic: 1
Backup Prepare Node	0
Apps Manager Database (Postgres)	Automatic: 0
Cloud Controller Database (Postgres)	Automatic: 0
UAA Database (Postgres)	Automatic: 0
UAA	Automatic: 1
Cloud Controller	Automatic: 1
HAProxy	0
Router	Automatic: 1
MySQL Monitor	Automatic: 1
Clock Global	Automatic: 1
Cloud Controller Worker	Automatic: 1
Diego BBS	Automatic: 1
Diego Brain	Automatic: 1
Diego Cell	Automatic: 3
Doppler Server	Automatic: 1
Loggregator Trafficcontroller	Automatic: 1
TCP Router	0
Push Apps Manager	Automatic: 1
Run Smoke Tests	Automatic: 1
Push Notifications	Automatic: 1
Run Notifications Tests	Automatic: 1

4. You can scale your deployment horizontally, by increasing the number of **Instances** of a job. You can also scale your deployment vertically, by adjusting 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, Elastic Runtime uses the recommended amount of resources for the job.

If you scale down or delete a job that uses persistent disk, Elastic Runtime marks the disk as “orphaned.” Orphaned disks are not attached to any job, and Elastic Runtime deletes them after five days. You can 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.

If you are using one of the following configurations, choose the values in the corresponding table to scale instances for your particular deployment:

- [External Databases](#)
- [Internal MySQL](#)
- [Internal Databases \(for Upgrades\)](#)
- [External Blobstore](#)
- [External Load Balancer](#)
- [JMX Bridge](#)

External Databases

If you are using an [external database](#), choose the following values in the Resource Config:

Job	Instance Count	Notes
MySQL Server	0	
MySQL Proxy	0	
Cloud Controller Database (Postgres)	0	
UAA Database (Postgres)	0	

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 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	Notes
MySQL Server	3	
MySQL Proxy	1	
Cloud Controller Database (Postgres)	0	
UAA Database (Postgres)	0	

Internal Databases (for Upgrades)

If you are upgrading from a previous installation that used both Postgres and MySQL databases, you must maintain this configuration to avoid data loss.

Job	Instance Count	Notes
MySQL Server	3	
MySQL Proxy	1	
Cloud Controller Database (Postgres)	1	
UAA Database (Postgres)	1	

External Blobstore

If you are using an external [Blobstore](#), choose the following value in the Resource Config:

Job	Instance Count	Notes
File Storage	0	

External Load Balancer

If you are using an external load balancer, choose the following values in the Resource Config:

Job	Instance Count	Notes
HAProxy	0	
Router	≥ 1	For Amazon Web Services, set the Elastic Load Balancer name in the Router's "External Load Balancer" field.
Diego Brain	≥ 1	For AWS, if you have the Diego SSH feature enabled, set the SSH ELB name in the Router's "External Load Balancer" field.

5. Choose the suggested values outlined in each scenario above, and click **Save**.

6. Return to the **Installation Dashboard** and click **Apply Changes**.

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 only running the MySQL cluster with a single node, you do not need to perform these steps.

Check the Health of Your Cluster

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

1. Use the 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 Elastic Runtime 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 validation-testing  
Creating org validation-testing as admin...  
OK  
  
Assigning role OrgManager to user admin in org validation-testing ...  
OK  
  
TIP: Use 'cf target -o validation-testing' 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 Elastic Runtime** tile.
- b. Click the **Status** tab.
- c. Record the IP addresses for all instances of the **MySQL Server** job.

5. Obtain the CCDB credentials for your MySQL server by performing the following steps:

- a. From the Elastic Runtime tile, click the **Credentials** tab.
- b. Locate the **Ccdb Credentials** entry in the **MySQL Server** section and click **Link to Credential**.
- c. Record the values for `identity` and `password`.

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 Elastic Runtime** 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 of CLI command:

```
$ cf delete-org data-integrity-test-organization
```

Using Docker Trusted Registries

Page last updated:

This topic describes how to configure your Docker Trusted Registries, such as Docker Hub, with Pivotal Cloud Foundry (PCF). Docker Trusted Registries are private Docker Registries that have a valid SSL certificate. To use Docker Trusted Registries, you must choose either to submit your root certificate authority (CA) certificate or provide the IP address for your Docker Trusted Registry.

Prerequisite: Ensure that you have enabled Docker support in PCF with the `cf enable-feature-flag diego_docker` command, as described in the [Using Docker in Cloud Foundry](#) topic.

Use a CA Certificate

If you provide your root CA certificate in the Ops Manager configuration, follow this procedure:

1. In the **PCF Ops Manager Installation Dashboard**, click the **Ops Manager Director** tile.
2. Click **Security**.

The screenshot shows the 'Security' tab selected in the 'Ops Manager Director' configuration interface. On the left sidebar, under 'AWS Config', 'Director Config', 'Create Availability Zones', 'Create Networks', 'Assign AZs and Networks', and 'Resource Config' are listed. Under 'Security', there is a 'Trusted Certificates' section containing a large redacted certificate block with the header '-----BEGIN CERTIFICATE-----'. Below it, a note states: 'These certificates enable BOSH-deployed components to trust a custom root certificate.' At the bottom of the page, there are two radio button options: 'Generate passwords' (selected) and 'Use default BOSH password'. A blue 'Save' button is located at the bottom center.

3. In the **Trusted Certificates** field paste one or more root CA certificates. The Docker Trusted Registry does not use the CA certificate itself but uses a certificate that is signed by the CA certificate.
4. Click **Save**.
5. If you are:
 - Configuring Ops Manager Installation for the first time, return to your specific IAAS configuration to continue the installation process.
 - Modifying an existing Ops Manager installation, return to the **PCF Ops Manager Installation Dashboard** and click **Apply Changes**.

After configuration, BOSH propagates this CA certificate to all application containers in your deployment. You can then push and pull images from your Docker Trusted Registries.

Use an IP Address Whitelist

If you choose not to provide a CA certificate, you must provide the IP address of your Docker Trusted Registry.

Note: Using a whitelist skips SSL validation. If you want to enforce SSL validation, enter the IP address of the Docker Trusted Registry in the **No proxy** field described [below](#).

1. Navigate to the PCF Operations Manager **Installation Dashboard**.
2. Click the **Pivotal Elastic Runtime** tile, and navigate to the **Application Containers** tab.

Enable microservice frameworks, private Docker registries, and other services that support your applications at a container level.

Enable Custom Buildpacks

Allow SSH access to app containers

Private Docker Insecure Registry Whitelist
 If you use private Docker image registries that are secured with self-signed certificates, enter them here as a comma-delimited list. List each registry as either an IP:Port tuple or a Hostname:Port tuple.

Docker Images Disk-Cleanup Scheduling on Cell VMs*

Never clean up Cell disk-space

Routinely clean up Cell disk-space

Clean up disk-space once threshold is reached

Threshold of Disk-Used (MB) (min: 1)*

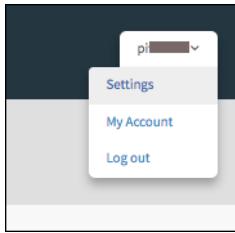
Save

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 are using 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 that app's space. Operators can grant those privileges in Apps Manager or via the cf CLI.
5. For **Private Docker Insecure Registry Whitelist**, provide the hostname or IP address and port that point to your Docker Trusted 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 following:
 - Never Cleanup Cell Disk-space
 - Routinely Cleanup Cell Disk-space
 - Cleanup disk-space once threshold is reached If you choose this option, enter the amount of disk space the Cell must reach before disk cleanup initiates under **Threshold of Disk-Used (MB)**.
7. Click **Save**.
8. If you are:
 - Configuring Elastic Runtime for the first time, return to your specific IaaS configuration to continue the installation process.
 - Modifying an existing Elastic Runtime installation, return to the PCF Ops Manager Installation Dashboard and click **Apply Changes**.

After configuration, Elastic Runtime allows Docker images to come through the specified IP address without checking certificates.

Set Proxies for Docker Trusted Registries

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 Trusted Registry, enter its IP address.
- **Https proxy**: If you have an HTTPS proxy server for your Docker Trusted Registry, enter its IP address.
- **No proxy**: If you do not use a proxy server, enter the IP address for the Docker Trusted Registry. This field may already contain proxy settings for the BOSH Director. Enter multiple IP addresses as a comma-separated list.

The screenshot shows the "Update Proxy Settings" pane within the "Settings" section of the Installation Dashboard. The pane has a title "Update Proxy Settings" and contains three input fields:

- "Http proxy": An input field containing the placeholder "Http proxy".
- "Https proxy": An input field containing the placeholder "Https proxy".
- "No proxy": An input field containing the value "10.10.10.4, 10.10.10.5".

Below the input fields is a blue "Update" button. On the left side of the pane, there is a sidebar with the following navigation links:

- Decryption Passphrase
- Authentication Method
- External API Access
- Proxy Settings** (this link is highlighted)
- Export Installation Settings
- Advanced

3. Click **Update**.

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 Elastic Runtime 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 Elastic Runtime**.
3. Click the **Custom Branding** tab.

Customize colors, images, and text for Apps Manager and the Cloud Foundry login portal.

Company Name

Accent Color

Main Logo (PNGs only)

Square Logo/Favicon (PNGs only)

Footer Text

Defaults to 'Pivotal Software Inc. All rights reserved.'

Add

Footer Links

You may configure up to three links in the Apps Manager footer

Classification Header/Footer Background Color

Classification Header/Footer Text Color

Classification Header Content

Classification Footer Content

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/Favicon**, enter a Base64-encoded URL string for a PNG image to use as your favicon, 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 **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.**

9. 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`.
10. 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 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 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.

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 Elastic Runtime**.
3. Click the **Apps Manager Config** tab.

Configure Apps Manager

Display Marketplace Service Plan Prices

Supported currencies as json *****
  Define the currency codes and associated symbols (defaults to `{"usd": "$", "eur": "€"}` if blank)

Product Name

Marketplace Name

Customize Sidebar Links
You may configure up to 10 links in the Apps Manager sidebar 

▶ Marketplace	
▶ Docs	
▶ Tools	

Save

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.

Monitoring App and Service Instance Usage

Page last updated:

This topic describes how to use the Cloud Foundry Command Line Interface (cf CLI) to retrieve usage information about your app 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.

Obtain System Usage Information

Before you can retrieve any app or service information, you must target the Cloud Controller and log in as admin, as follows:

1. Target the endpoint of your Cloud Controller.

```
$ cf api api.YOUR-DOMAIN
```

2. Log in with your credentials.

```
$ cf login -u admin
API endpoint: api.YOUR-DOMAIN
Email: user@example.com
Password:
Authenticating...
OK
...
Targeted org YOUR-ORG
Targeted space development
API endpoint: https://api.YOUR-DOMAIN (API version: 2.52.0)
User:      user@example.com
Org:       YOUR-ORG
Space:    development
```

3. Run `curl` for the `/system_usage_report` on the Usage service.

```
$ curl "https://app-usage.YOUR-DOMAIN/system_usage_report" -k -v -H "authorization: `cf oauth-token`"
```

Obtain Usage Information about an Org

To obtain individual org usage information, use the following procedure. You must log in as an admin or as an Org Manager or Org Auditor for the org you want to view.

1. Run `cf login -u USERNAME`.
2. Run `curl` for the `/app_usages` or `/service_usages` endpoints on the Usage service.

```
$ curl "https://app-usage.YOUR-DOMAIN/organizations/" cf org 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": [
    {
      "app_guid": "00ecd7ce-1dd0-4b3f-63b9-744c9de42afc",
      "app_name": "your-app",
      "duration_in_seconds": 76730,
      "instance_count": 6,
      "memory_in_mb_per_instance": 64,
      "space_guid": "44435fd6-fbac-5049-bbfc-92d1603a5e98",
      "space_name": "outer-space"
    }
  ]
}
```

Or run the following:

```
$ curl "https://app-usage.YOUR-DOMAIN/organizations/'cf org YOUR-ORG --guid'/service_usages?start=YYYY-MM-DD&end=YYYY-MM-DD" -k -v -H "authorization: `cf oauth-to
{
  "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: "c6445eb3-fdac-4049-bafc-94d1703d5e78"
      space_name: "outer-space"
    }
  ]
}
```

Retrieve Apps Information

- The `apps` endpoint retrieves 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`.

2. The `summary` endpoint under each app-specific URL retrieves 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-DOMAIN.io"
      }
    },
    ...
  ],
  "running_instances": 5,
  "services": [
    {
      "guid": "b9cdr456-3c61-4f8a-a307-9bf836fb2e",
      "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": "fbccc3af-3e8d-4ee7-adfe-3f12a137ed66",
        "name": "turtle",
        "service": {
          "guid": "34dbc753-34ed-4cf1-9a87-a255dfca5339b",
          "label": "elephantsql",
          "provider": null,
          "version": null
        }
      }
    }
  ]
}
```

3. 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](#).

Retrieve Services Information

Use `cf curl` to retrieve service instance information. The `service_instances?` endpoint retrieves details about both bound and unbound service instances:

```
$ 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"
      }
    }
  ]
}
```

Monitoring Instance Usage with Apps Manager

Page last updated:

This topic describes how to retrieve app and service instance usage information using Apps Manager.

You can also retrieve app 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 and Service Instance Usage](#) topic.

There are two ways to monitor app 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.

Accounting Report					
Monthly count of App Instances in use for all orgs (not including the system org)					
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 memory usage 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. Next to the org **QUOTA**, click **Usage Report**.

ORG
QUOTA
[Usage Report](#)

random-org-name

0 MB / 10 GB
0%

Space (1)
Domain (1)
Members (2)
Settings

The top of the Usage Report displays total app memory usage and service instance usage by all spaces in your org.

Usage Report

Period
December 1st - Current
[Download Zip](#)

ORG	APP INSTANCE HOURS	APP MEMORY	SERVICE INSTANCE HOURS
bhale	288.0 hrs	144.0 GB hrs	0.0 hrs

The report displays your app memory usage and service instance usage by spaces.

SPACE	APP INSTANCE HOURS	APP MEMORY	SERVICE INSTANCE HOURS	
development	288.0 hrs	144.0 GB hrs	0.0 hrs	
APPS(1)	INSTANCE COUNT AVG	MAX	INSTANCE HOURS TOTAL	MEMORY USAGE TOTAL
apps-manager-...	1.7	2	288.0 hrs	144.00 GB hrs
SERVICES (0)	PLAN		INSTANCE HOURS	

Deploying Diego for Windows

Page last updated:

This topic contains instructions for setting up a Windows cell in a Diego deployment. For more information about Diego, see the [Diego Architecture](#) topic.

A **cell** is a virtual machine (VM) that stages, hosts, and manages application lifecycles. You can install a Windows cell into your Pivotal Cloud Foundry (PCF) deployment by connecting directly to a Windows VM.

Limitations

Unsupported Features

Diego for Windows does not yet support the following features:

- Guaranteed binary compatibility with Diego BOSH releases
- BOSH rolling updates for PCF or core operating system updates in Windows
- Fair sharing of CPU resources

 **Note:** Diego for Windows does not enforce any CPU limits.

- Container SSH access from the Cloud Foundry Command Line Interface (cf CLI)
- ICMP egress by default. You must explicitly enable ICMP through security groups.
- Emitting firewall logs into the CF log pipeline

Stability and Scalability Expectations

Capacity planning for Windows instances of PCF varies greatly based on the overhead caused by the components you add to the instance.

Supported Applications

The following application types are known to run correctly on Diego Windows:

- [ASP.NET MVC](#)

 **Note:** Twelve-factor ASP.NET MVC apps compiled against .NET 3.5+ were tested most extensively.

- [Windows-compiled executables](#)
- [Batch scripts with a manually specified start command](#)
- [WCF Applications](#)

Install

Prerequisites

- A working Diego deployment
- A Windows Server 2012R2 VM instance that is routable to your Diego deployment
 - See [recommended instance types](#) in the GitHub Diego release repo for details.
 - If you are creating a new Windows image, and not using one predefined and supplied by your IaaS, we recommend using this [ISO image](#).

- as a starting point. You must have an MSDN account to download this ISO image.
- We recommend at least 50 gigabytes of storage space for your Windows VM instance.

Note: The IP address of your Windows cell must not conflict with the IP addresses of VMs managed by BOSH. To prevent a conflict, use separate subnets in your VPC for the Windows cell and BOSH VMs, or assign the Windows cell an IP address from the [Excluded IP Range](#) that you declared in Ops Manager.

Step 1: Retrieve Setup Files

Perform the following steps to download the necessary setup files:

- From your Windows cell, navigate to the [Elastic Runtime product on Pivotal Network](#).
- Select the **DiegoWindows** file group from the table.

The screenshot shows the 'Elastic Runtime' product page. In the 'Release Download Files' section, there is a list of files under the 'DiegoWindows 1.6.0' folder. The 'DiegoWindows 1.6.0' folder is highlighted with a green arrow pointing to it. The files listed are: PCF 1.6 Documentation (23.3 MB), PCF 1.6 Cloudformation script for AWS (12.6 KB), PCF Elastic Runtime (3.34 GB), and the 'DiegoWindows 1.6.0' folder itself (5 Files).

- Download the **setup.ps1** and **generate.exe** files. Keep this window open to complete the steps below.

Note: If you download the **generate.exe** file using Internet Explorer, Internet Explorer removes the `.exe` extension. You must rename the file to add the `.exe` extension.

Step 2: Configure Windows Cell

Perform the following steps to configure your Windows cell:

- Using **File Explorer**, navigate to the location where you downloaded the **setup.ps1** and **generate.exe** files.
- Right-click on the **setup.ps1** file and select **Run with PowerShell**. The **setup.ps1** script configures Windows features, DNS settings, and the firewall for your Windows cell.

Note: Some IaaSes may require elevated Windows privileges to run the `setup.ps1` script. If you receive a `PSecurity Unauthorized Access` error, use the `Set-ExecutionPolicy Unrestricted` PowerShell cmdlet before re-running the `setup.ps1` script.

Step 3: Download Your Manifest

Perform the following steps to download your Cloud Foundry manifest:

1. SSH into your Ops Manager VM. The steps vary depending on your IaaS. For more information, see the [SSH into Ops Manager](#) section of the *Advanced Troubleshooting with the BOSH CLI* topic.
2. From your Ops Manager VM, use the BOSH CLI to target and log in to your BOSH Director. The steps vary depending on whether your PCF deployment uses internal authentication or an external user store. For more information, see the [Log into BOSH](#) section of the *Advanced Troubleshooting with the BOSH CLI* topic.
3. Use the `bosh deployments` command to list your deployments:

```
$ bosh deployments
Acting as user 'director' on 'p-bosh'
RSA 1024 bit CA certificates are loaded due to old openssl compatibility
+-----+-----+-----+
| Name | Release(s) | Stemcell(s) | Cloud Config |
+-----+-----+-----+
| cf-222e11e1111111e1111 | cf-autoscaling/36 | bosh-google-kvm-ubuntu-trusty-go_agent/3263.7 | none |
|   | cf-mysql/26.6 | | |
|   | cf239.0.26 | | |
|   | cflinuxfs2-rootfs/1.33.0 | | |
|   | consul/108.0.2 | | |
|   | diego/0.1485.1 | | |
|   | etcd/60.0.1 | | |
|   | garden-runc/0.9.2 | | |
|   | ipsec/1.5.37 | | |
|   | mysql-backup/1.25.0 | | |
|   | mysql-monitoring/6 | | |
|   | notifications-ui/17 | | |
|   | notifications/24 | | |
|   | pivotal-account/1 | | |
|   | push-apps-manager-release/652 | | |
|   | routing/0.138.6 | | |
|   | service-backup/14 | | |
+-----+-----+-----+
```

4. Review the output and identify the name of your Cloud Foundry deployment. In the above example, the name is `cf-222e11e1111111e1111`.
5. Use the `bosh download` command to download the manifest of your Cloud Foundry deployment as `cf.yml`:

```
$ bosh download manifest cf-222e11e1111111e1111 cf.yml
Acting as user 'director' on deployment 'cf-222e11e1111111e1111' on 'p-bosh'
RSA 1024 bit CA certificates are loaded due to old openssl compatibility
Deployment manifest saved to 'cf.yml'
```

6. Copy the `cf.yml` file from the Ops Manager VM to your Windows cell in one of two ways:
 - From your Windows cell, use [WinSCP](#) to copy the manifest from the Ops Manager VM.
 - From your local Mac or Linux machine, use `scp` to copy the manifest from the Ops Manager VM, then use a Remote Desktop Protocol (RDP) client like [Microsoft Remote Desktop](#) to mount a directory containing the manifest on your local machine as a drive on the Windows cell. For more information, see the [Microsoft documentation](#).

Step 4: Run Install Script Generator

From your Windows cell, run `generate.exe` with the following arguments:

- **manifest:** The path to the manifest file downloaded from your Ops Manager BOSH Director
- **outputDir:** The directory that will contain the required certificates and a script to run the installers

```
$ generate.exe ^
-manifest /tmp(cf.yml) ^
-outputDir C:\diego-windows
```

 **Note:** The parameters for `generate.exe` are case-sensitive.

Step 5: Install MSI

1. Download **DiegoWindows.msi** and **GardenWindows.msi** from the same Pivotal Network file group to the `outputDir` that you specified

above.

Release Download Files / DiegoWindows 1.6.0		Release Details
GardenWindows.msi	4.67 MB 1.6.0	RELEASE DATE 2015-10-26
setup.ps1	5.85 KB 1.6.0	RELEASE TYPE Minor Release
DiegoWindows.msi	11.8 MB 1.6.0	END OF GENERAL SUPPORT 2017-03-31
AWS CloudFormation	8.96 KB 1.6.0	
generate.exe	7.43 MB 1.6.0	

Release Details

RELEASE DATE: 2015-10-26
RELEASE TYPE: Minor Release
END OF GENERAL SUPPORT: 2017-03-31

RELEASE DESCRIPTION
This release includes Diego, the new runtime container orchestration technology for Elastic Runtime, among many other new features.

DEPENDS ON
Products in the "Depends On" section must be installed or upgraded to Elastic Runtime 1.6.0. Please upgrade these products to one of the listed versions.

Ops Manager

Release Download Files / DiegoWindows 1.6.0		Release Details
GardenWindows.msi	4.67 MB	2015-10-26
setup.ps1	5.85 KB	Minor Release
DiegoWindows.msi	11.8 MB	2017-03-31
AWS CloudFormation	8.96 KB	
generate.exe	7.43 MB	

Release Details

2015-10-26
Minor Release
2017-03-31

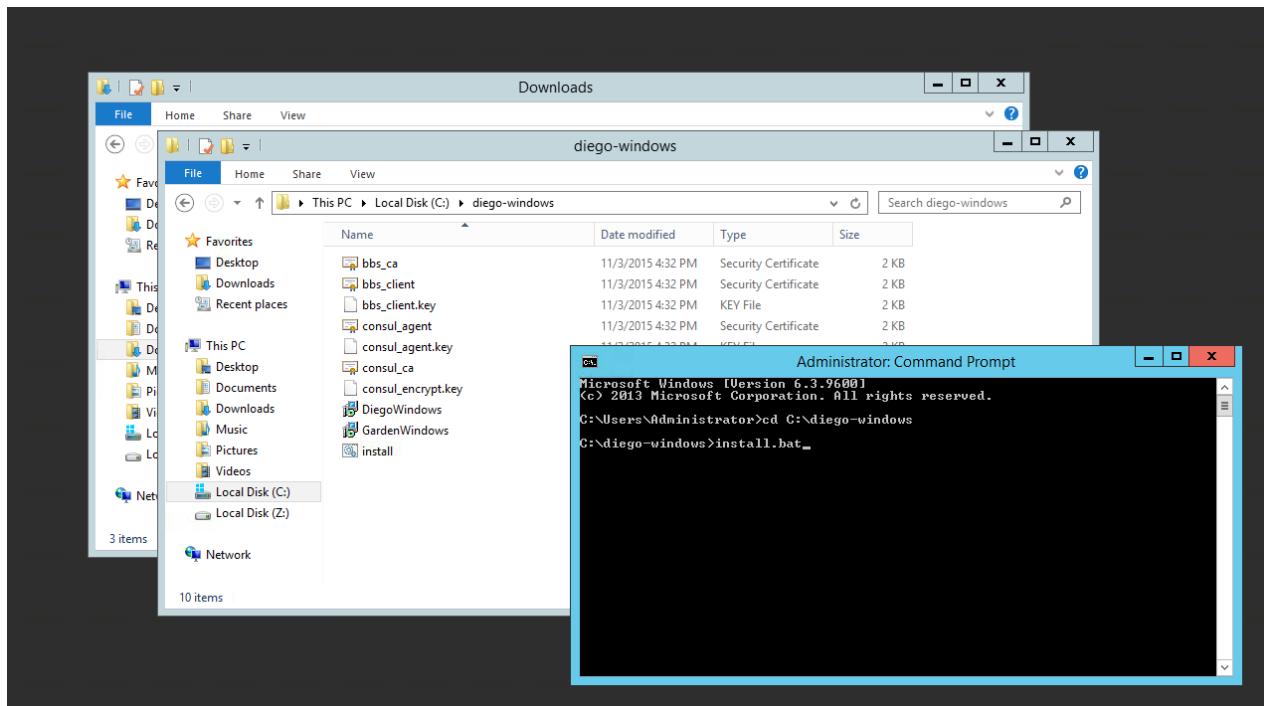
This release includes Diego, the new runtime container orchestration technology for Elastic Runtime, among many other new features.

Products in the "Depends On" section must be installed prior to upgrading to Elastic Runtime 1.6.0. Please install or upgrade these products to one of the listed versions.

Do you want to run or save GardenWindows.msi (4.67 MB) from dtb5pzswc1e.cloudfront.net?
This type of file could harm your computer.

- From a command prompt, run `install.bat` from the `outputDir`.

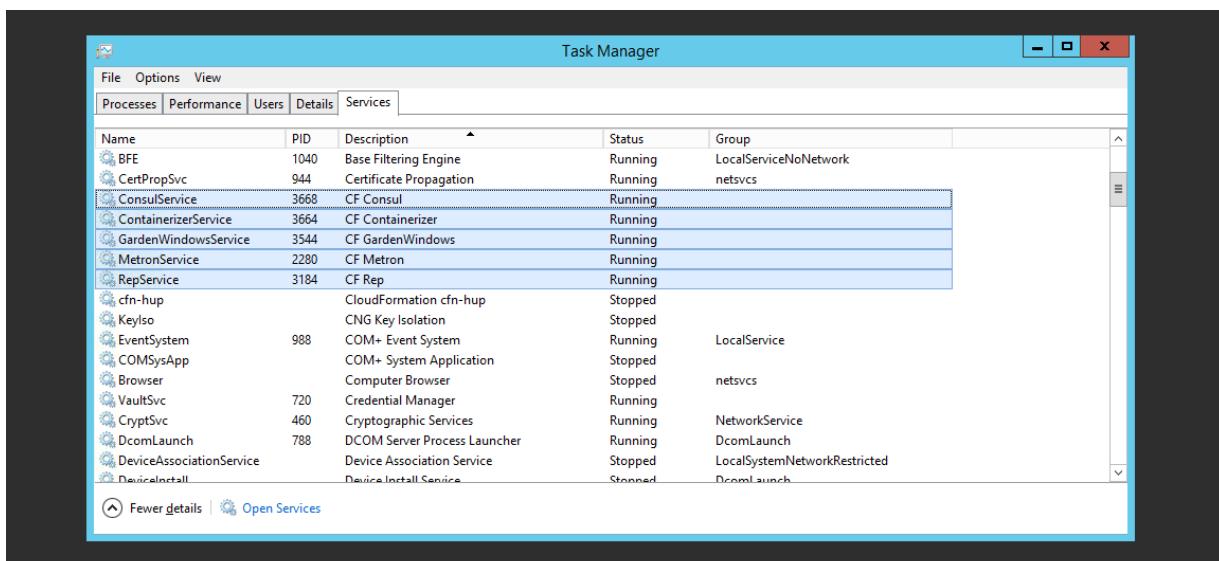
Note: By default, Containerizer stores container files at `C:\containerizer`. To modify the default, open `install.bat` and set `CONTAINER_DIRECTORY` to the directory where you want Containerizer to store container files.



Step 6: Confirm Successful Deployment

Follow the steps below to deploy a sample .NET application to one of your Windows cells and exercise basic Elastic Runtime functionality to ensure that your deployment functions properly.

1. Launch **Task Manager**.
2. Navigate to the Services tab. Confirm that the following five services are running:
 - ConsulService
 - ContainerizerService
 - GardenWindowsService
 - MetronService
 - RepService



3. Clone the [CF Smoke Tests](#) repository.
4. Follow the instructions from the CF Smoke Tests README to run the smoke tests against your environment with the `enable_windows_tests` configuration flag set to `true`.

Operating Diego for Windows

Page last updated:

This topic describes how to operate a Diego deployment on Windows. For instructions on setting up a Windows Diego deployment, see the [Deploying Diego for Windows](#) topic.

Customize Cells

Pivotal recommends that you keep customization of Windows cells to a minimum. If you do customize your cells, you must apply any software or configuration settings to every cell in your cluster.

 The Cloud Foundry team tests against a clean Windows Server 2012 R2 installation, following the [standard install instructions](#). Software and configuration not included in this clean install, such as those applied by Group Policy, can cause complex and unknown interactions with Cloud Foundry.

Reboot Cells

Before rebooting a Windows cell, you must first trigger an [evacuation](#) to avoid application downtime.

To trigger an evacuation, execute the following PowerShell script:

```
Set-Service RepService -startuptype "Disabled"

Invoke-WebRequest -Uri http://localhost:1800/evacuate -Method Post

while ($true) {
    try {
        Get-WebRequest "http://localhost:1800/ping"
    } catch {
        [system.exception]
        break;
    }
}

Set-Service RepService -startuptype "Automatic"
```

Retrieve Version Numbers

To retrieve a version number for an executable or MSI, right-click the file and click **Properties**.

To retrieve the version number for the `setup.ps1` script, pass the `-version` flag on the command line:

```
> powershell .\setup.ps1 -version
```

Custom CA Certificates

If your applications require custom CA certificates in order to communicate with other components, install the certificates on the Windows cell. Applications running on the cell will trust certificates that the local machine or domain trust.

See the [Manage Trusted Root Certificates](#) TechNet article for information.

Upgrade a Cell

Diego retains backwards compatibility with Windows cells, which allows for rolling upgrades. Greenhouse/.NET implements a cell evacuation prior to new releases to support upgrades.

To upgrade a Windows cell, perform the following steps:

1. Spin up a new cell.
2. Trigger an evacuation on an old cell using the PowerShell script from the [Rebooting Cells](#) section above.
3. Shut down the old cell when the evacuation completes.
4. Repeat until all cells are updated.

Monitoring a Pivotal Cloud Foundry Deployment

Page last updated:

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 [Datadog](#) or [OpenTSDB](#).

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 Elastic Runtime 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 Firehose logs and metrics come from two sources: the BOSH Health Monitor and Cloud Foundry components.

BOSH Health Monitor

The BOSH layer that underlies PCF generates [healthmonitor](#) metrics for all VMs in the deployment. The Pivotal Cloud Ops team considers the following of these to be the most important for monitoring system health:

- `bosh.healthmonitor.system.cpu`: CPU usage, percent of total available on VM
- `bosh.healthmonitor.system.mem`: Memory usage, percent of total available on VM
- `bosh.healthmonitor.system.disk`: Disk usage, percent of total available on VM
- `bosh.healthmonitor.system.healthy`: `1` if VM is healthy, `0` otherwise

Cloud Foundry Components

Cloud Foundry component VMs for executive control, hosting, routing, traffic control, authentication, and other internal functions generate metrics. See the [Cloud Foundry Component Metrics](#) topic for a detailed list of metrics generated by Cloud Foundry.

The Pivotal Cloud Ops team considers the following PCF Component metrics often useful to monitor:

- auctioneer.AuctioneerFetchStatesDuration
- auctioneer.AuctioneerLRPAuctionsFailed
- bbs.Domain.cf_apps
- bbs.CrashedActualLRPs
- bbs.LRPsMissing
- bbs.ConvergenceLRPDuration
- bbs.RequestLatency
- DopplerServer.listeners.receivedEnvelopes
- DopplerServer.TruncatingBuffer.totalDroppedMessages
- gorouter.total_routes
- gorouter.ms_since_last_registry_update
- MetronAgent.dropsondeMarshaller.sentEnvelopes
- nsync_bulker.DesiredLRPSyncDuration
- rep.CapacityRemainingMemory
- rep.CapacityTotalMemory
- rep.RepBulkSyncDuration
- route_emitter.RouteEmitterSyncDuration

PCF components specific to your IaaS also generate key metrics for health monitoring.

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 Elastic Runtime, 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](#) repo.

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. If you are not using the JMX Bridge nozzle, install the [HM Forwarder](#) process to run on the BOSH Health Monitor VM. This process routes health metrics to the local Metron agent, and it does not install automatically as part of PCF. You do not need the HM Forwarder with the JMX Bridge nozzle, which queries the Health Monitor directly.
3. Install a custom app to generate smoke test or other custom system metrics.
4. 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.

If you do not use the [JMX Bridge](#) [OpenTSDB](#) Firehose Nozzle, you must install a BOSH HM Forwarder job on the VM that runs the BOSH Health Monitor and its Metron agent.

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.

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 %>}) > 1",
  "message": "##Description:\nDiego internal metrics\n## Escalation Path:\nDiego\n## Possible Causes:\nThose alerts were a pretty strong signal for us to look at the BBS, which was locked up\n## Pot. 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:

- [Datadog](#)
- [OpenTSDB](#) alerts and [Grafana](#) dashboard
- [New Relic](#)
- [vRealize Operations \(vROPS\)](#)
- [AppDynamics](#)

Providing a Certificate for Your SSL/TLS Termination Point

Page last updated:

This topic describes how to configure SSL/TLS termination for HTTP traffic in [Pivotal Cloud Foundry](#) (PCF) Elastic Runtime with an SSL certificate, as part of the process of configuring Elastic Runtime for deployment.

About SSL/TLS Termination in PCF Elastic Runtime

When you deploy PCF, you must configure the SSL/TLS termination for HTTP traffic in your Elastic Runtime configuration. You can terminate SSL/TLS at any one of the following entry points into PCF:

- Gorouter
- Load balancer and Gorouter
- Load balancer
- HAProxy

The following table summarizes SSL/TLS termination options in PCF and provides guidance on which option to choose for your deployment.

If the following applies to you:	Then terminate SSL/TLS at:	Related topic and configuration procedure:
<ul style="list-style-type: none"> • You want the most performant and recommended option, and • You can use a single SAN certificate for your deployment for your wildcard domains. 	Gorouter only. Select the Forward SSL to Elastic Runtime Router in your Elastic Runtime network configuration.	Terminating SSL/TLS at the Gorouter Only
<ul style="list-style-type: none"> • You cannot use a single SAN certificate for all system and application domains for your deployment, or • You require Extended Validation (EV) certificates. 	Load balancer only. Select the Forward unencrypted traffic to Elastic Runtime Router	Terminating SSL/TLS at the Load Balancer
<ul style="list-style-type: none"> • You want a higher level of security, and • You do not mind a slightly less performant deployment, and • You can use a single SAN certificate on the Gorouter, either for all system and application domains (but with a different key than for the same certificates hosted on your load balancer) or for a single domain (but with hostname verification disabled on the load balancer). 	Load balancer and Gorouter. Select the Forward SSL to Elastic Runtime Router in your Elastic Runtime network configuration.	Terminating SSL/TLS at Gorouter and Load Balancer
<ul style="list-style-type: none"> • You can use a single SAN certificate for your deployment for your wildcard domains, and • You are deploying a test or development environment, or • You do not mind a slightly less performant deployment. 	HA Proxy. Select the Forward SSL to HA Proxy .	Terminating SSL/TLS at HA Proxy

Certificate Requirements for PCF

The following list describes the certificate requirements for deploying PCF.

- You must obtain at least one SSL/TLS certificate for your environment.
 - In a production environment, use a signed SSL/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, or use the Elastic Runtime Ops Manager interface to [generate a certificate](#) for you.

 **Note:** Certificates generated in Elastic Runtime 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 PCF must be encoded in the PEM format.
- When you generate a certificate, save the private key used to generate your certificate in a safe place. You must upload its contents to the Elastic Runtime networking configuration.
- 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. For this reason, if EV certificates are required, then terminate TLS/SSL at the [load balancer only](#) for the same reason stated above. You can avoid having to reissue and reinstall certificates on the Gorouter for every app or UAA security zone.

Certificate Requirements on AWS

If you are deploying PCF on AWS, then the certificate that you configure in Elastic Runtime must match the certificate that you upload to AWS as a prerequisite to PCF deployment. For more information, see the [Upload an SSL Certificate](#) section of the [Deploying the CloudFormation Template for PCF on AWS](#) topic.

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 (Elastic Runtime 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 Elastic Runtime Router** in your Elastic Runtime network configuration. This point-of-entry selection accommodates this special characteristic of GCP deployments.

Creating a Wildcard Certificate for PCF Deployments

This section describes how to create or generate a certificate for your PCF Elastic Runtime 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 SSL certificates.

- You can create a self-signed certificate, or
- You can have [Elastic Runtime 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:

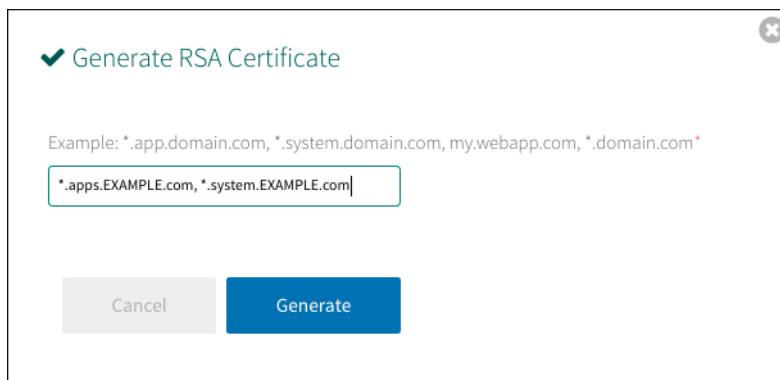
- Specify your registered wildcard domain as the `Common Name`. For example, `*.yourdomain.com`.
- If you are using a split domain setup that separates the domains for `apps` and `system` components (recommended), then enter the following values in the `Subject Alternative Name` of the certificate:
 - `*.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:** SSL 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 Elastic Runtime

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the **Elastic Runtime** tile in the Installation Dashboard.
3. Click **Networking**.
4. Click **Generate RSA Certificate** to populate the **Router SSL Termination Certificate and Private Key** 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 an ephemeral 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.

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 **Elastic Runtime** tile.
3. Click **Advanced Features**.
4. Under **Enabling NFSv3 volume services**, select **Enable**.

BETA: Enabling NFSv3 volume services will allow application developers to bind existing NFS volumes to their applications for shared file access.*

- Enable
 Disable

5. Click **Save**.
6. Return to the Ops Manager Installation Dashboard and click **Apply Changes** to redeploy.
7. 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.

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 creates a single isolation segment.

For more information about how isolation segments work, see the [Isolation Segments](#) section of the *Understanding Cloud Foundry Security* topic. After installing the PCF Isolation Segment tile, see the [Managing Isolation Segments](#) topic for more information about managing an isolation segment.

Step 1: (Optional) Configure Routing

By default, the Elastic Runtime Router handles traffic for your isolation segment. However, you can deploy a dedicated router for your isolation segment instead.

To deploy a dedicated router, perform the following steps:

1. Add a load balancer in front of the Elastic Runtime Router. The steps to do this depend on your IaaS.
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

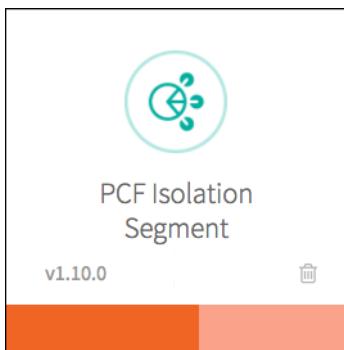
Perform the following steps to install the PCF Isolation Segment tile:

1. Download the product file from [Pivotal Network](#).
2. Click **Import a Product** and select the downloaded product file.
3. Under **PCF Isolation Segment** in the left column, click the plus sign.

Step 3: Configure the Tile

Perform the following steps to configure the PCF Isolation Segment tile:

1. Click the orange **PCF Isolation Segment** tile to start the configuration process.



2. Click **Assign AZs and Networks**.

AZ and Network Assignments

Place singleton jobs in

- us-central1-a
- us-central1-b
- us-central1-c

Balance other jobs in

- us-central1-a
- us-central1-b
- us-central1-c

Network

pcf-network

Save

3. Select an availability zone for your singleton jobs, and one or more availability zones to balance other jobs in.
4. Select a network. This network does not need to be the same network where you deployed Elastic Runtime. 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.
5. Click **Save**.
6. Click **Application Containers**.

Enable custom configuration that supports your applications at a container level.

Private Docker Insecure Registry Whitelist

10.10.10.10:8888,example.com:8888

Segment Name *

my_segment

Save

7. (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`.
8. Under **Segment Name**, enter the name of your isolation segment. This name must be unique across your PCF deployment. You use this name when managing the isolation segment.
9. Click **Networking**.

Configure security and routing services for your isolation segment.

Router IPs

Applications Subnet (Only change this if you need to avoid address collision with a third-party service on the same subnet.) *

10.254.0.0/22

Applications Network Maximum Transmission Unit (MTU) (in bytes) *

1454

10. (Optional): Under **Router IPs**, enter one or more static IP addresses for the routers that handles this isolation segment. These IP addresses must be within the subnet CIDR block that you defined in the Ops Manager network configuration. If you have a load balancer, configure it to point to these IP addresses.
11. (Optional): Under **Applications Subnet**, enter a CIDR subnet mask specifying the range of available IP addresses to assign to your app containers. This must be different from the network used by the system VMs. Only modify the default value if you need to avoid address collision with a third-party service on the same subnet.
12. (Optional): Under **Applications Network Maximum Transmission Unit**, change the Maximum Transmission Unit (MTU). The default is 1454 bytes.
13. Under **TLS Termination Certificates**, select one of the following options:
 - **Forward SSL to Isolation Segment Router with provided certificates:** Select this option if your deployment uses an external load balancer that can forward encrypted traffic to the Elastic Runtime Router for the isolation segment, or if you are running a development environment that does not require load balancing. Complete the fields for the **Router SSL Termination Certificate and Private Key** and **Router SSL Ciphers**.
 - **Forward SSL to Isolation Segment Router with ERT Router certificates:** Select this option to inherit the certificates provided to the Elastic Runtime Router when you configured the Elastic Runtime tile. This option assumes an external load balancer is configured to forward encrypted traffic.
 - **Forward unencrypted traffic to Elastic Runtime Router:** Select this option if your deployment uses an external load balancer that cannot forward encrypted traffic to the Elastic Runtime Router, or for a development environment that does not require load balancing.

TLS Termination Certificates*

- Forward SSL to Isolation Segment Router with provided certificates. Assumes an external load balancer is configured to forward encrypted traffic.

Router SSL Termination Certificate and Private Key *

Certificate PEM

Private Key PEM

[Generate RSA Certificate](#)
Required for SSL encryption between the load balancer and router(s).

Router SSL Ciphers

Forward SSL to Isolation Segment Router with ERT Router certificates. Assumes an external load balancer is configured to forward encrypted traffic. Will inherit the certificates provided to the Router in the Elastic Runtime Tile.

Forward unencrypted traffic to Elastic Runtime Router. Assumes an external load balancer is configured to forward unencrypted traffic.

- Click **Save**.
- (Optional): Edit the configurations in **Advanced Features** as desired.
- If you are using a dedicated router for your isolation segment, click **Resource Config** and enter the wildcard DNS entry attached to your load balancer into the **Router** row under **Load Balancers**.

Resource Config

JOB	INSTANCES	PERSISTENT DISK TYPE	VM TYPE	LOAD BALANCERS	INTERNET CONNECTED
Router	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)	*.iso.exam	<input checked="" type="checkbox"/>
Diego Cell	Automatic: 3	None	Automatic: xlarge.disk (cpu: 4, ram: 16 GB, disk: 128)		<input checked="" type="checkbox"/>

Save

- (Optional): Edit the **Stemcell** configuration as desired.
- 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.

Administering and Operating Cloud Foundry

For Administrators of a Running Cloud Foundry Deployment

- [Managing Custom Buildpacks](#)
- [Adding a Custom Stack](#)
- [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](#)
- [Application Security Groups](#)
- [Administering Container-to-Container Networking](#)
- [Managing Isolation Segments](#)
- [Feature Flags](#)

For Operators Deploying Cloud Foundry

- [Managing Diego Cell Limits During Upgrade](#)
- [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](#)

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 an administrator for your Cloud Foundry org 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 buildpack, 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 upgrading 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 / unlock
```

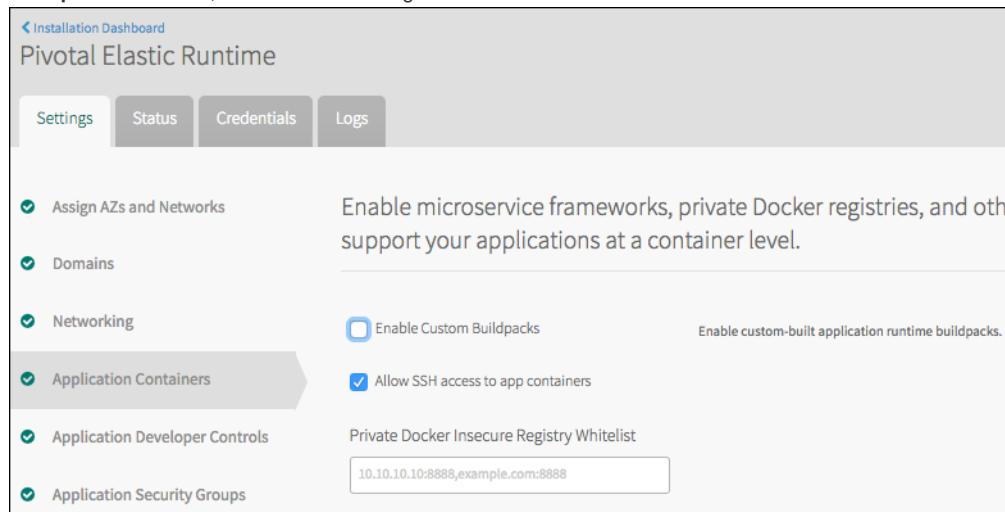
flag lets you continue to use your existing buildpack even after you upgrade. Locked buildpacks are not updated when Cloud Foundry updates. You must unlock them manually to update them.

```
cf update-buildpack BUILDPACK [-p PATH] [-i POSITION] [--enable|--disable] [-l 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 Elastic Runtime 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 Elastic Runtime. 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.

Adding a Custom Stack

Page last updated:

This topic outlines how add a custom stack under Diego architecture. To add a stack, you first build a BOSH job template that installs the stack on the host machine. Then you configure your deployment manifests so that Cloud Foundry can run the job when it creates cells.

The [Cloud Foundry Stacks](#) repository contains scripts for building your own custom stacks, as well as the available Cloud Foundry stacks.

The following example adds a new Linux-based `pancakes` stack for use with the [Garden-runC](#) operating system. This `pancakes` stack could, for example, support applications that require an old version of CentOS or Ubuntu.

Step 1: Create a BOSH Job Template

Stacks exist in a subdirectory on their host machine, typically under `/var/vcap/packages` or `/var/vcap/data`. Your BOSH job template must deploy the stack onto a host machine, and provide lifecycle binaries that work with your stack. The lifecycle binaries for your stack are helper programs that stage and run apps on the stack file system. To create a `pancakes-release` job template that deploys a custom stack, follow these steps:

1. Create a BOSH release [pancakes-release](#) for a job template that expands a stack into place in its subdirectory. For example, a `pancakes-rootsfs` template might create a full Linux root file system in the directory `/var/vcap/packages/pancakes-rootsfs/rootsfs`. See the ['rootfses' job template in diego-release](#) for one way to do this.
2. Create lifecycle binaries for your stack. See the [diego-release](#) repository for examples of app lifecycle binary source code:
 - [Buildpack App Lifecycle](#)
 - [Docker App Lifecycle](#)
 - [Windows App Lifecycle](#)
3. Generate a gzipped tar archive of the lifecycle binaries, `pancakes-app-lifecycle.tgz`.
4. Create a dummy `pancakes-app-lifecycle` job template as a package within `pancakes-release`. Include the `pancakes-app-lifecycle.tgz` file in the job template directory.

 **Note:** The `pancakes-app-lifecycle` job template does not need to run any process of its own.

5. List the dummy `pancakes-app-lifecycle` job as a dependency in the `pancakes-release` spec file. This makes BOSH publish the lifecycle binaries to `/var/vcap/packages` for inclusion in any cells that use the `pancakes` stack.

Step 2: Update the Manifests

1. Add the `pancakes-rootsfs` job and release name to the Diego manifest, to the list of job templates defined for the `cell` object under `base_job_templates`. This makes the expanded rootsfs available locally on the Diego cell, at `/var/vcap/packages/pancakes-rootsfs/rootsfs`. For example, in the manifests generated with the [spiff-based tooling](#) in [diego-release](#), add the lines shown in **bold** to the following list of cell job templates:

```
cell:
  - name: rep
    release: diego
  - name: consul_agent
    release: cf
  - name: garden
    release: garden-runc
  - name: rootfses
    release: diego
  - name: pancakes-rootsfs
    release: pancakes
  - name: metron_agent
    release: cf
```

2. Add `pancakes-app-lifecycle` to the `base_job_templates` list under the `file_server` Diego job. In [diego-release](#), the `file_server` job resides in the `access` job template group. For example, add the lines shown in **bold** to the following list of job templates:

```
access:
  - name: ssh_proxy
    release: diego
```

```

- name: consul_agent
  release: cf
- name: metron_agent
  release: cf
- name: file_server
  release: diego
- name: pancakes-app-lifecycle
  release: pancakes

```

3. The `diego.rep.preloaded_rootfses` property of the Cell Rep holds an array associating stacks with their file system root locations. Add a pair to this list to associate the `pancakes` stack with its file system root location, set up on the cell by the `pancakes-rootfs` job. For example, in the `preloaded_rootfses:` property under `rep:` in your [Diego manifest](#), set the array to the following by adding the text shown **in bold**:

```

["cflinuxfs2:/var/vcap/packages/cflinuxfs2/rootfs",
 "pancakes:/var/vcap/packages/pancakes-rootfs/rootfs"]

```

4. Configure the stager and nsync components to use the `pancakes` lifecycle binary bundle to start and stop apps running on the `pancakes` stack. For example, in [CAPI release](#), add the line shown in **bold** to the `default` list under the manifest definitions for both `diego.nsnc.lifecycle_bundles` and `diego.stager.lifecycle.bundles`:

```

description: "List of lifecycle bundles arguments for different stacks in form 'lifecycle-name:path/to/bundle'"
default:
  - "buildpack/cflinuxfs2:buildpack_app_lifecycle/buildpack_app_lifecycle.tgz"
  - "buildpack/pancakes:pancakes-app-lifecycle/pancakes-app-lifecycle.tgz"
  - "buildpack/windows2012R2:windows_app_lifecycle/windows_app_lifecycle.tgz"
  - "docker:docker_app_lifecycle/docker_app_lifecycle.tgz"

```

5. Configure the Cloud Controller for the new stack by adding it to the 'cc.stacks' property in the CF manifest. For example, in the [diego-release](#) manifest generation stubs for CF, add the lines shown in **bold**:

```

properties:
  cc:
    stacks:
      - name: "cflinuxfs2"
        description: "Cloud Foundry Linux-based filesystem"
      - name: "windows2012R2"
        description: "Windows Server 2012 R2"
      - name: "pancakes"
        description: "Linux-based filesystem, with delicious pancakes"

```

Using Docker in Cloud Foundry

Page last updated:

This topic provides information about how Docker works in Cloud Foundry and describes how to push an application with a Docker image. For information about Diego, see the [Diego Architecture](#) topic.

In addition to the default Linux environment provided by Cloud Foundry, operators can specify a Docker image that provides the file system used by the container. When pushing an app, you can specify the location of the docker image you want to use.

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. Earlier versions of Diego used Garden-Linux. For more information, see the [Garden](#) topic.

 **Note:** Elastic Runtime versions v1.8.8 and above use Garden-runC instead of Garden-Linux.

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.

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.

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

 **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.

To allow Docker-based applications to run, a Cloud Controller administrator can enable the `diego_docker` feature flag with the following command:

```
$ cf enable-feature-flag diego_docker
```

To disallow Docker-based applications, a Cloud Controller administrator can run the following command:

```
$ 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.

Push a Docker Image with the cf CLI

Follow these instructions to deploy updated or new Docker images using the [Cloud Foundry Command Line Interface \(cf CLI\)](#).

When pushing an app, you specify either a Docker image on Docker Hub or the URI to a trusted Docker registry. Diego does not support using private registries. Also, when using Diego, you must use a registry that supports the [Docker Registry API V2](#).

Push a Docker Image Using Docker Hub

Run `cf push test-app -o cloudfoundry/MY-DOCKER-IMAGE` to deploy a Docker image. Replace `MY-DOCKER-IMAGE` with the name of an image from an accessible Docker Registry.

For example, the following command pushes the `test-app` on Docker Hub to Cloud Foundry:

```
$ cf push test-app -o cloudfoundry/test-app
```

Push a Docker Image Using Docker Trusted Registries

To deploy a Docker image using Docker trusted registries, run:

```
$ cf push my-app -o MY-PRIVATE-REGISTRY.DOMAIN:5000/image/name:v2
```

Replace `MY-PRIVATE-REGISTRY.DOMAIN` with your domain name, which resolves to the private registry. This example uses the following values:

- `5000`: The port on which your private registry serves traffic.
- `image/name`: The name of the Docker image repository.
- `v2`: A tag in the specified Docker image repository that refers to a specific image.

Docker Volume Support

Diego now 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](#).

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 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

Page last updated:

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 [Elastic Runtime > 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	<code>cf create-user USERNAME PASSWORD</code>	<code>cf create-user Alice pa55w0rd</code>
Delete a user	<code>cf delete-user USERNAME</code>	<code>cf delete-user Alice</code>

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	<code>cf orgs</code>	<code>cf orgs</code>
View all users in an organization by role	<code>cf org-users ORGANIZATION-NAME</code>	<code>cf org-users my-example-org</code>
Assign an org role to a user	<code>cf set-org-role USERNAME ORGANIZATION-NAME ROLE</code>	<code>cf set-org-role Alice my-example-org OrgManager</code>
Remove an org role from a user	<code>cf unset-org-role USERNAME ORGANIZATION-NAME ROLE</code>	<code>cf unset-org-role Alice my-example-org OrgManager</code>

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	<code>cf spaces</code>
View all users in a space by role	cf space-users ORGANIZATION-NAME SPACE-NAME	<code>cf space-users my-example-org development</code>
Assign a space role to a user	cf set-space-role USERNAME ORGANIZATION-NAME SPACE-NAME ROLE	<code>cf set-space-role Alice my-example-org development SpaceAuditor</code>
Remove a space role from a user	cf unset-space-role USERNAME ORGANIZATION-NAME SPACE-NAME ROLE	<code>cf unset-space-role Alice my-example-org development SpaceAuditor</code>

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-abed123
```

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`, `uaa.admin`, and `scim.read`.

```
$ uaac member add cloud_controller.admin_read_only Bob
$ uaac member add uaa.admin 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
```

- 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
```

- 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.

- 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.
- 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`.
- 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"`
- Retrieve the name of your SAML provider by navigating to the Elastic Runtime 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 Elastic Runtime](#) topic.
- 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

- 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.
- Run `cf login -u NEW-ADMIN-USERNAME -p NEW-ADMIN-PASSWORD` to log in.
- Run `cf create-user NEW-USER-NAME NEW-USER-PASSWORD` to create a new user.

```
$ cf login -u Adam -p newAdminSecretPassword
```

```
$ cf create-user Charlie aNewPassword
```

Change Passwords

- 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.
- 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
```

- 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: aBedEfg0hIJKlm123.c
token_type: bearer
expires_in: 43200
scope: uaa.admin clients.secret password.read
jti: 91b3-abcd1233
```

- 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.
- If the admin user lacks permissions to change passwords, add the permissions by following these steps:
 - Run `uaac client update admin --authorities "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 --authorities "'uaac client get admin | \
awk '/:/{e=0}/authorities:/{{e=1;if(e==1){$1=""}}}'" password.write"

$ uaac token delete
$ uaac token client get admin
```

- 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
```

- 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:

- Run `uaac target uaa.YOUR-DOMAIN` to target your UAA server.

```
$ uaac target uaa.example.com
```

- Record the `uaa:admin:client_secret` from your deployment manifest.

- 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
```

- 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
```

- 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.

- 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
```

- 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
```

- 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 <code>starting</code> .	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.	<code>true</code> or <code>false</code>	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.	<code>true</code> or <code>false</code>	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 an org in one of two ways:

- Directly [modify the CF deployment manifest](#) before deploying
- Use [cf create-quota](#) after deploying

Modify the CF Deployment Manifest

The CF Deployment manifest specifies the default quota plans applied to orgs. Follow the steps below to modify the default quota plans by locating and editing the manifest.

1. In a terminal window, run `bosh edit deployment` to open the deployment manifest YAML file in your default text editor.
2. Search for `quota_definitions`.
3. Add a new quota definition with values that you specify. Use the default quota definition as a formatting template. The following example shows the `quota_definitions` portion of the `cf.yml` manifest after adding the `silver_quota` plan:

```
quota_definitions:  
  default:  
    memory_limit: 1024M  
    non_basic_services_allowed: true  
    total_routes: 1000  
    total_services: 100  
    trial_db_allowed: true  
  silver_quota:  
    memory_limit: 2048M  
    non_basic_services_allowed: true  
    total_routes: 500  
    total_services: 25  
    trial_db_allowed: true
```

4. Save and close the deployment manifest.
5. Run `bosh deploy` to apply the change.

 **Note:** Any subsequent updates to the quota in the manifest are not applied to your environment after the initial deploy, even though BOSH updates the controller instance with new settings.

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

You can modify an existing quota plan for an org in one of two ways:

- Directly [modify the CF deployment manifest](#) before deploying.
- Use `cf update-quota` after deploying.

Modify the Manifest

1. In a terminal window, run `bosh edit deployment` to open the deployment manifest YAML file in your default text editor.
2. Search for `quota_definitions`.
3. Modify the value of the attribute.
4. Save and close the deployment manifest.

Use cf update-quota

1. Run `cf quotas` to find the names of all quota definitions available to your org. Note 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. Note 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

You must have the following setup before using the Notifications Service:

- Install [Elastic Runtime](#).
- You must have `admin` permissions on your Cloud Foundry instance. You also must configure [Application Security Groups \(ASGs\)](#).
- Install the `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 need to create [UAA](#) scopes:

1. Use `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. 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": false, "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.

Application Security Groups

Page last updated:

This page assumes you are using cf CLI v6.4 or later.

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.

Application Security Groups

Application Security Groups (ASGs) are 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.

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.

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.

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 permitting 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

Elastic Runtime 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, Elastic Runtime 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 the CC API. The cf CLI command supports bind-security-group space-scoped running ASGs, but not space-scoped staging ASGs.

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
<code>protocol</code>	<code>tcp</code> , <code>udp</code> , <code>icmp</code> , or <code>all</code>	Required
<code>destination</code>	A single IP address, an IP address range like <code>192.0.2.0-192.0.2.50</code> , or a CIDR block to allow network access to	
<code>ports</code>	A single port, multiple comma-separated ports, or a single range of ports that can receive traffic. Examples: <code>443</code> , <code>80,8080,8081</code> , <code>8080-8081</code>	Required when <code>protocol</code> is <code>tcp</code> or <code>udp</code>
<code>code</code>	ICMP code	Required when <code>protocol</code> is <code>icmp</code>
<code>type</code>	ICMP type	Required when <code>protocol</code> is <code>icmp</code>
<code>log</code>	Set to <code>true</code> 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.	
<code>description</code>	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 [Procedures for Working with ASGs](#) below.

Task	For more information, see
1. Review the existing ASGs. If this is a new deployment, probably these are the Default ASGs alone.	View ASGs
2. Create new ASGs.	Create ASGs
3. Update the existing ASGs.	Update ASGs
4. Bind ASGs to an ASG set.	Bind ASGs
5. If you need to delete ASGs, unbind and delete them.	Unbind ASGs Delete ASGs

Procedures for Working with ASGs

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
<code>cf security-groups</code>	All ASGs
<code>cf staging-security-groups</code>	All ASGs applied to the platform-wide staging ASG set
<code>cf running-security-groups</code>	All ASGs applied to the platform-wide running ASG set
<code>cf security-group SECURITY-GROUP</code>	All rules in the ASG named <code>SECURITY-GROUP</code> , for example, <code>cf security-group dns</code>

 **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": 1  
  },  
  {  
    "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](#) [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 the following Cloud Controller (CC) API commands:

```
GET /v2/security_groups/:guid/staging_spaces  
PUT /v2/spaces/:guid/staging_security_groups/:security_group_guid data  
PUT /v2/security_groups/:guid/staging_spaces/:space_guid  
DELETE /v2/spaces/:guid/staging_security_groups/:security_group_guid data  
DELETE /v2/security_groups/:guid/staging_spaces/:space_guid
```

These API calls require administrator access. Additionally, the payload returned from API `GET` calls to `/v2/spaces/` and `/v2/spaces/:guid` includes a link to the `staging_security_groups_url`.

For more information about using these CC API commands, see the [Cloud Foundry API](#) documentation.

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](#) of 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](#) of 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`. 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.

Example:

```
$ cf unbind-security-group my-asg my-org my-space
```

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
<code>dns</code>	DNS, either public or private
<code>public-networks</code>	Public networks, excluding IaaS metadata endpoints
<code>private-networks</code>	Private networks in accordance with RFC-1918
<code>load-balancers</code>	The internal Elastic Runtime load balancer and others
<code>internal-proxies</code>	Internal proxies
<code>internal-databases</code>	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 bind-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>

Administering Container-to-Container Networking

This topic describes how to enable and use the Container-to-Container Networking feature. For an overview of how Container-to-Container Networking works, see the [Understanding Container-to-Container Networking](#) topic.

Enable Container-to-Container Networking

This section explains how to enable container-to-container networking in PCF. You enable container-to-container networking as an **Advanced Feature** in the Ops Manager Elastic Runtime tile. Container-to-Container networking is currently in beta; it is not supported and not guaranteed to work as expected when running with all possible combinations of PCF services.

1. In Ops Manager, navigate to the **Installation Dashboard > Elastic Runtime** tile.
2. Click **Advanced Features**.
3. In the **Advanced Features** pane, select **Enable Container-to-Container Networking**.

Enabling the Container-to-Container network will allow policy-driven, direct communication between containers on an overlay network occupying the 10.255.0.0/16 range.*

- Enable Container-to-Container Networking
 Disable Container-to-Container Networking

Save

4. Click **Save**.
5. Return to the Installation Dashboard.

Create Policies for Container-to-Container Networking

This section describes how to create and modify Container-to-Container Networking policies using a plugin for the CF CLI.

To use the plugin, you must have the `network.admin` UAA scope. This scope gives you the right to create a policy between any two apps in your CF deployment. Depending on the security structure of your organization, you can either assign this scope to developers so that they can create their own policies or you can have your developers send you requests. For more information, see [Creating and Managing Users with the UAA CLI \(UAAC\)](#).

Install the Plugin

Follow these steps to download and install the Network Policy plugin for the CF CLI:

1. Download the `network-policy-plugin` for your operating system from the [Container-to-Container Networking Release repository](#).
2. To change the permissions of the plugin file and complete the installation, enter the following commands:

```
$ chmod +x ~/Downloads/network-policy-plugin  
$ cf install-plugin ~/Downloads/network-policy-plugin
```

Create a Policy

To create a policy that allows direct network traffic from one app to another, enter the following command:

```
$ cf allow-access SOURCE-APP DESTINATION-APP --protocol PROTOCOL --port PORT
```

Replace the placeholders in the above command as follows:

- `SOURCE-APP` is the name of the app that will be sending traffic.
- `DESTINATION-APP` is the name of the app that will be receiving traffic.
- `PROTOCOL` is one of the following: `tcp` or `udp`.
- `PORT` is the port at which to connect to the destination app. The allowed range is 1 to 65535.

The following example command allows access from the `frontend` app to the `backend` app over TCP at port 8080:

```
$ cf allow-access frontend backend --protocol tcp --port 8080
Allowing traffic from frontend to backend as admin...
OK
```

List Policies

You can list all the policies in your deployment or just the policies for which a single app is either the source or the destination:

- To list the all the policies in your deployment, enter the following command:

```
$ cf list-access
```

- To list the policies for an app, enter the following command:

```
$ cf list-access --app MY-APP
```

The following example command lists policies for the app `frontend`:

```
$ cf list-access --app frontend
Listing policies as admin...
OK

Source  Destination  Protocol  Port
frontend  backend    tcp      8080
```

Delete a Policy

To delete a policy that allows direct network traffic from one app to another, enter the following command:

```
$ cf remove-access SOURCE-APP DESTINATION-APP --protocol PROTOCOL --port PORT
```

For example,

```
$ cf remove-access frontend backend --protocol tcp --port 8080
Denying traffic from frontend to backend as admin...
OK
```

Managing Isolation Segments

Page last updated:

This topic describes how operators can isolate deployment workloads to dedicated resource pools called isolation segments.

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.

To manage the isolation segment, an operator sends `curl` requests to the Cloud Controller API (CAPI) endpoint with the GUID of the isolation segment and an authorization token. For more information, see the [Identifying the API Endpoint for your Elastic Runtime Instance](#) topic.

Operators can perform the following operations on isolation segments:

- [List](#) isolation segments
- [List orgs](#) or [spaces](#) in an isolation segment
- [Retrieve](#) an isolation segment object
- [Update](#) an isolation segment
- [Delete](#) an isolation segment
- [Add orgs](#) to isolation segments
- [Remove orgs](#) from isolation segments
- [Set the default](#) isolation segment for an org
- [Add and remove spaces](#) to isolation segments

Isolation Segment Contents

An isolation segment object in the Cloud Controller Database (CCDB) includes the following:

- The unique name of the isolation segment
- A unique GUID
- Timestamps for the object's creation and most recent update
- Links to API endpoints for isolation segment requests

In the CCDB, an isolation segment object does not identify the orgs and spaces that it includes. Instead, the org and space objects define this relationship by including the GUIDs of the isolation segments that they belong to.

List Isolation Segment Information

The `curl` requests listed in the sections below retrieve information for the isolation segments that you have access to, filtered by [parameters](#) that you can include. The isolation segments you can see information for depends on your role, as follows:

- **Admins** see all isolation segments in the system.
- **Other users** see the isolation segments that their orgs have been added to.

List Isolation Segments

The following request returns a filtered list of the isolation segment objects that you can access:

```
curl "https://api.example.org/v3/isolation_segments" \
-X GET \
-H "Authorization: bearer 7h15154l0n64lph4num3r1c57rln6"
```

The list returns as a `resources` property structured in a paginated format.

List Orgs for an Isolation Segment

The following example request returns a list of orgs in the isolation segment with the given GUID. The request only returns orgs that you can access:

```
curl "https://api.example.org/v3/isolation_segments/323f211e-fca3-4161-9bd1-615392327913/relationships/organizations" \
-X GET \
-H "Authorization: bearer 7h15154l0n64lph4num3r1c57r1n6"
```

The request returns a list of orgs in the following format:

```
HTTP/1.1 200 OK

{
  "data": [
    {
      "name": "my_org",
      "guid": "45a66ed9-cb76-46c3-92dd-b29187b50bfb",
      "link": "/v2/organizations/45a66ed9-cb76-46c3-92dd-b29187b50bfb"
    },
    {
      "name": "my_other_org",
      "guid": "d0540a63-3bcc-42ff-abd9-8a30328ba296",
      "link": "/v2/organizations/d0540a63-3bec-42ff-abd9-8a30328ba296"
    }
  ]
}
```

List Spaces for an Isolation Segment

The following example request returns a list of spaces in the isolation segment with the given GUID. The request only returns spaces that you can access:

```
curl "https://api.example.org/v3/isolation_segments/323f211e-fca3-4161-9bd1-615392327913/relationships/spaces" \
-X GET \
-H "Authorization: bearer 7h15154l0n64lph4num3r1c57r1n6"
```

The request returns a list of spaces in the following format:

```
HTTP/1.1 200 OK

{
  "data": [
    {
      "name": "my_space",
      "guid": "68d54d31-9b3a-463b-ba94-e8e4c32edbac",
      "link": "/v2/spaces/68d54d31-9b3a-463b-ba94-e8e4c32edbac"
    },
    {
      "name": "my_other_space",
      "guid": "b19f6525-cbd3-4155-b156-dc0c2a431b4c",
      "link": "/v2/spaces/b19f6525-cbd3-4155-b156-dc0c2a431b4c"
    }
  ]
}
```

Retrieve an Isolation Segment

To retrieve an isolation segment by its GUID, send a `curl` command like the following to the `/isolation_segments/GUID` endpoint of your CAPI:

```
curl "https://api.example.org/v3/isolation_segments/323f211e-fca3-4161-9bd1-615392327913" \
-X GET \
-H "Authorization: bearer 7h15154l0n64lph4num3r1c57r1n6"
```

This example request returns the [contents](#) of the isolation segment:

HTTP/1.1 200 OK

```
{  
  "guid": "323f211efea3-4161-9bd1-615392327913",  
  "name": "my_segment",  
  "created_at": "2016-10-19T20:25:04Z",  
  "updated_at": "2016-11-08T16:41:26Z",  
  "links": {  
    "self": {  
      "href": "https://api.example.org/v3/isolation_segments/323f211efea3-4161-9bd1-615392327913"  
    },  
    "spaces": {  
      "href": "https://api.example.org/v3/isolation_segments/323f211efea3-4161-9bd1-615392327913/relationships/spaces"  
    },  
    "organizations": {  
      "href": "https://api.example.org/v3/isolation_segments/323f211efea3-4161-9bd1-615392327913/relationships/organizations"  
    }  
  }  
}
```

Manage an Isolation Segment

The `curl` requests listed in the sections below make changes to isolation segment objects in the CCDB. Only admins can make these changes.

Update an Isolation Segment

The following example renames the isolation segment with the given GUID to `my_isolation_segment`:

```
curl "https://api.example.org/v3/isolation_segments/323f211efea3-4161-9bd1-615392327913" \  
-X PUT \  
-H "Authorization: bearer 7h15154l0n64lph4num3r1c57rln6" \  
-d'{  
  "name": "my_isolation_segment"  
}'
```

The request returns the following output:

HTTP/1.1 200 OK

```
{  
  "guid": "323f211efea3-4161-9bd1-615392327913",  
  "name": "my_isolation_segment",  
  "created_at": "2016-10-19T20:25:04Z",  
  "updated_at": "2016-11-08T16:41:26Z",  
  "links": {  
    "self": {  
      "href": "https://api.example.org/v3/isolation_segments/323f211efea3-4161-9bd1-615392327913"  
    },  
    "spaces": {  
      "href": "https://api.example.org/v3/isolation_segments/323f211efea3-4161-9bd1-615392327913/relationships/spaces"  
    },  
    "organizations": {  
      "href": "https://api.example.org/v3/isolation_segments/323f211efea3-4161-9bd1-615392327913/relationships/organizations"  
    }  
  }  
}
```

Delete an Isolation Segment

The following example deletes an isolation segment with the given GUID:

```
curl "https://api.example.org/v3/isolation_segments/323f211efea3-4161-9bd1-615392327913" \  
-X DELETE \  
-H "Authorization: bearer 7h15154l0n64lph4num3r1c57rln6" \  
-
```

The request outputs the following:

HTTP/1.1 204 No Content

Manage Isolation Segment Relationships

The `curl` requests listed in the sections below add and remove orgs and spaces from isolation segments.

Add Orgs to an Isolation Segment

Only admins can add orgs to isolation segments.

In the data field of the `curl` command, specify one or more orgs by GUID to add to the isolation segment. The following example adds two orgs to an isolation segment:

```
curl "https://api.example.org/v3/isolation_segments/323f211efea3-4161-9bd1-615392327913/relationships/organizations" \
-X POST \
-H "Authorization: bearer 7h15154l0n64lph4num3r1c57rln6" \
-d'{
  "data": [
    { "guid":"45a66ed9-cb76-46c3-92dd-b29187b50bfb" },
    { "guid":"d0540a63-3bec-42ff-abd9-8a30328ba296" }
  ]
}'
```

The request outputs the following:

HTTP/1.1 201 OK

```
{
  "guid": "323f211efea3-4161-9bd1-615392327913",
  "name": "my_segment",
  "created_at": "2016-10-19T20:25:04Z",
  "updated_at": "2016-11-08T16:41:26Z",
  "links": {
    "self": {
      "href": "https://api.example.org/v3/isolation_segments/323f211efea3-4161-9bd1-615392327913"
    },
    "spaces": {
      "href": "https://api.example.org/v3/isolation_segments/323f211efea3-4161-9bd1-615392327913/relationships/spaces"
    },
    "organizations": {
      "href": "https://api.example.org/v3/isolation_segments/323f211efea3-4161-9bd1-615392327913/relationships/organizations"
    }
  }
}
```

If an org is entitled to 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.

Remove Orgs from an Isolation Segment

The following example removes two orgs from an isolation segment:

```
curl "https://api.example.org/v3/isolation_segments/323f211efea3-4161-9bd1-615392327913/relationships/organizations" \
-X DELETE \
-H "Authorization: bearer 7h15154l0n64lph4num3r1c57rln6" \
-d'{
  "data": [
    { "guid":"45a66ed9-cb76-46c3-92dd-b29187b50bfb" },
    { "guid":"d0540a63-3bec-42ff-abd9-8a30328ba296" }
  ]
}'
```

The request outputs the following:

HTTP/1.1 204 No Content

 **Note:** You cannot remove an org from an isolation segment if the isolation segment contains a space within that org or if it is the default isolation segment for that org.

Set a Default Isolation Segment for an Org

Only admins and org managers can set the default isolation segment of an org.

When an org has a default isolation segment, new spaces created within the org will be in this default isolation segment unless specified otherwise.

You set the default isolation segment for an org by sending a request to the endpoint for the organization, setting its `default_isolation_segment_guid` data property to the GUID of the new default isolation segment. For example:

```
curl "https://api.example.org/v2/organizations/45a66ed9-cb76-46c3-92dd-b29187b50bfb" \
-X PUT \
-H "Authorization: bearer 7h15154l0n64lph4num3r1c57rln6" \
-d'{
  "default_isolation_segment_guid":"323f211efea3-4161-9bd1-615392327913"
}'
```

Add or Remove Spaces in an Isolation Segment

Only admins and org managers can add or remove space in an isolation segment.

You add a space to an isolation segment by sending a request to the endpoint for the space, setting its `isolation_segment_guid` data property to the GUID of the new default isolation segment. For example:

```
curl "https://api.example.org/v2/spaces/68d54d31-9b3a-463b-ba94-e8e4c32edbac" \
-X PUT \
-H "Authorization: bearer 7h15154l0n64lph4num3r1c57rln6" \
-d'{
  "isolation_segment_guid":"323f211efea3-4161-9bd1-615392327913"
}'
```

For an example of how to remove a space from an isolation segment, see the following command:

```
curl "https://api.example.org/v2/spaces/68d54d31-9b3a-463b-ba94-e8e4c32edbac/isolation_segment" -X DELETE
```

Feature Flags

Page last updated:

This topic describes how Cloud Foundry (CF) administrators can set feature flags using the cf Command Line Interface (cf CLI) to enable or disable 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
```

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`, `diego_docker`, and `task_creation`. 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. This feature is under development. 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.

For more information about feature flag commands, see the **Feature Flags** section of the [Cloud Foundry API documentation](#).

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 on 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	No limit set	No	100% of total instances	No

	NO INITIAL SET	NO	10% OF TOTAL INSTANCES	NO
1.8.29				
PCF 1.8.30 +	200	Yes	10% of total instances	No
PCF 1.9.0 to 1.9.7	No limit set	No	10% of total instances	Yes
PCF 1.9.8 +	200	Yes	10% of total instances	Yes
PCF 1.10.0 and later	200	Yes	10% 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 Elastic Runtime tile.

1. Log in to Operations Manager.
2. Click the Elastic Runtime tile.
3. Click **Application instances** in the sidebar.
4. In the Max Inflight Container Starts field, type the maximum number of started instances.

The screenshot shows the 'Application Instances' settings page. On the left, a sidebar lists various application components: Application Containers (selected), Application Developer Controls, Application Security Groups, Authentication and Enterprise SSO, Databases, Internal MySQL, File Storage, System Logging, Custom Branding, Apps Manager, and Email Notifications. The 'Max Inflight Container Starts' field is located in the main content area, which is divided into sections for Docker Registry Whitelist, Docker Image Disk-Cleanup Scheduling, and Disk Usage Threshold. The 'Max Inflight Container Starts' field contains the value '200', which is highlighted with a red border. Below the field is a 'Save' button.

Allow SSH access to app containers
Enable SSH when an app is created
Private Docker Insecure Registry Whitelist 10.10.10.10:8888,example.com:8888
Docker Images Disk-Cleanup Scheduling on Cell VMs*
Never clean up Cell disk-space
Routinely clean up Cell disk-space
Clean up disk-space once threshold is reached
Threshold of Disk-Used (MB) (min: 1) *
10240
Max Inflight Container Starts *
200

5. Click **Save**.

Enabling IPv6 for Hosted Applications

Page last updated:

The procedure described below allows apps deployed to Elastic Runtime to be reached using IPv6 addresses.

 **Note:** Amazon Web Services (AWS) EC2 instances currently do not support IPv6.

Elastic Runtime 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.

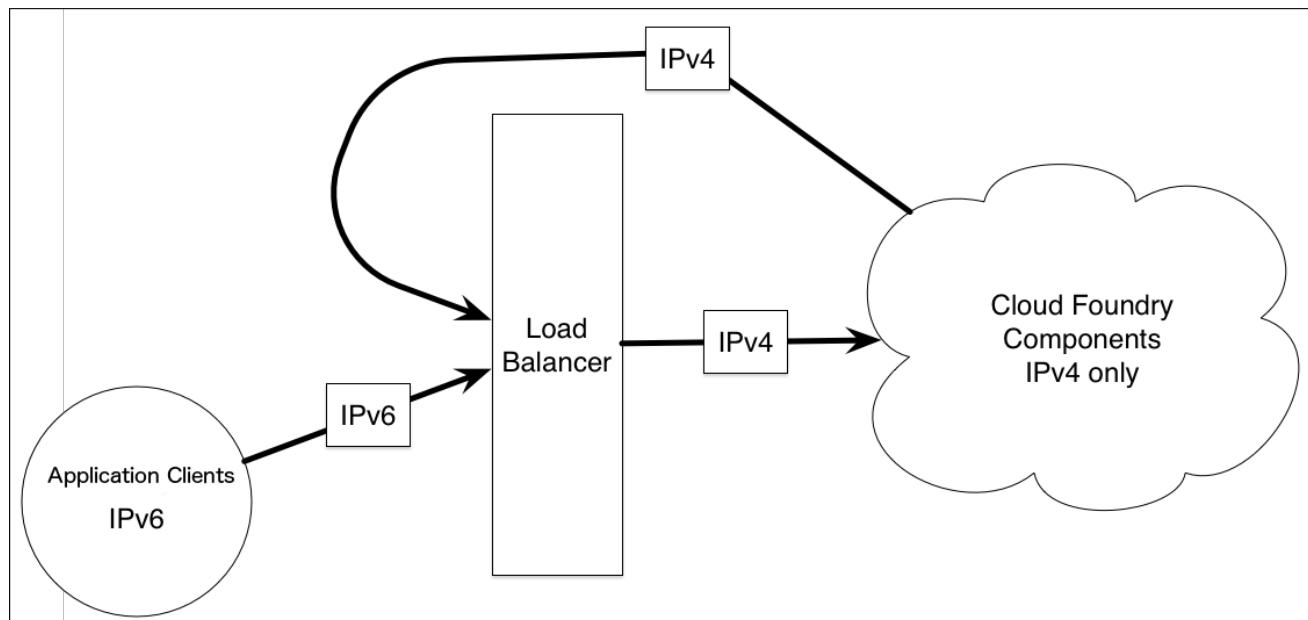
Complete the following steps to enable support for IPv6 application domains:

1. Set up an external load balancer for your Elastic Runtime 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 Elastic Runtime installation.



See [Routes and Domains](#) for more information about domains in Elastic Runtime.

Securing Traffic into Cloud Foundry

Page last updated:

This topic describes the options for securing HTTP traffic into your Elastic Runtime deployment with SSL/TLS certificates. You can configure the location where your deployment terminates TLS depending on your needs and certificate restrictions.

Protocol Support

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, terminate TLS at your load balancer or at the application. See [TCP Routing](#) for details.

SSL/TLS Termination Options

There are three options for terminating SSL/TLS for HTTP traffic. You can terminate TLS at the Gorouter, your load balancer, or at both.

The following table summarizes SSL/TLS termination options and which option to choose for your deployment.

If the following applies to you:	Then configure SSL/TLS termination at:	Related topic and configuration procedure
<ul style="list-style-type: none"> You want the most performant and recommended option, and You can use a single SAN certificate for all system and application domains for your deployment. 	Gorouter only	Terminating SSL/TLS at the Gorouter Only
<ul style="list-style-type: none"> You cannot use a single SAN certificate for all system and application domains for your deployment, or You require Extended Validation (EV) certificates. 	Load Balancer only	Terminating SSL/TLS at the Load Balancer Only
<ul style="list-style-type: none"> You want a higher level of security, and You do not mind a slightly less performant deployment, and You can use a single SAN certificate on the Gorouter, either for all system and application domains (but with a different key than for the same certificates hosted on your load balancer) or for a single domain (but with hostname verification disabled on the load balancer). 	Load Balancer and Gorouter	Terminating SSL/TLS at the Load Balancer and Gorouter

Certificate Requirements

The following requirements apply to the certificates you use to secure traffic into Elastic Runtime

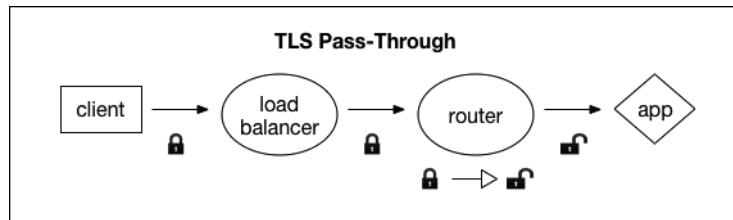
- In a production environment, use a signed SSL/TLS certificate (trusted) from a known certificate authority (CA).
- In a development or testing environment, you can use a trusted CA certificate or a self-signed certificate generated with `openssl` or a similar tool.
- The Gorouter currently only supports configuring a single HTTPS certificate. The certificate on the Gorouter must be associated with all the domains it may receive requests for so that HTTPS can validate the request. To associate multiple domains to a single certificate, including wildcard domains, use [Subject Alternative Name \(SAN\)](#), an X.509 extension.
- If wildcard certificates are not supported for some or all of your domains, then terminate TLS at your load balancer.
- If you cannot use a SAN certificate, terminate TLS at your load balancer.
- Extended Validation (EV) certificates support multiple hostnames, like SAN, but do not support wildcards. For this reason, if EV certificates are required, then terminate TLS at the load balancer only.
- If you must terminate TLS at your load balancer, you may optionally secure requests from the load balancer to the Gorouter by terminating TLS at the Gorouter also. To accomplish this without using a SAN certificate on the Gorouter, you will have to disable hostname validation on the load balancer, as the domain in the certificate hosted by the Gorouter will not match requests forwarded to it for applications on Elastic Runtime by the load balancer.

Terminating SSL/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.

Certificate Guidelines

- The Gorouter currently only supports configuring a single HTTPS certificate.
- Use a SAN certificate because Elastic Runtime requires multiple wildcard domains.

About HTTP Header Forwarding

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 client, and `X-Forwarded-Proto` to the scheme of the client request.

If you terminate TLS/SSL at the Gorouter only, you do not need to configure any additional HTTP header forwarding on your load balancer.

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 requests from the client to the Gorouter.
2. Navigate to the Ops Manager Installation Dashboard.
3. Click the **Elastic Runtime** tile in the Installation Dashboard.
4. Click **Networking**.
5. Under **Select one of the following point-of-entry options** select the first option, **Forward SSL to Elastic Runtime Router**.
6. Enter your PEM-encoded certificate and your PEM-encoded private key in the fields under **Router SSL Termination Certificate and Private Key**. You can either upload your own certificate or generate an RSA certificate in Elastic Runtime. For options and instructions on creating a certificate for your wildcard domains, see [Creating a Wildcard Certificate for PCF Deployments](#)
7. If you want to use a specific set of SSL ciphers for the Router, configure **Router SSL Ciphers**. Enter a colon-separated list of custom SSL ciphers to pass to the router. Otherwise, leave this field blank.
8. 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.

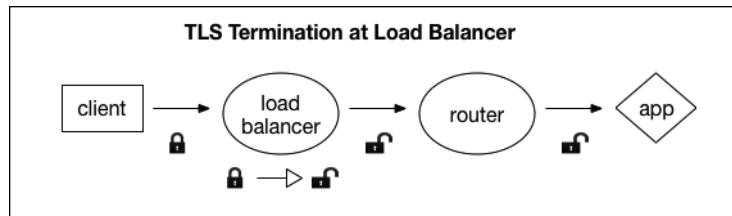
9. Click **Save**.

Terminating SSL/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.



Certificate Guidelines

You can use multiple certificates on your load balancer if the load balancer supports multiple VIPs or SNI.

About HTTP Header Forwarding

If you terminate SSL/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.

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 Elastic Runtime 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 **Elastic Runtime** tile in the Installation Dashboard.
 4. Click **Networking**.
 5. For PCF deployments on OpenStack or vSphere, enter the load balancer IP address that you used to set up a wildcard DNS record in the **Router IPs** field. For more information, see the Elastic Runtime networking configuration topic for [OpenStack](#) or [vSphere](#).
 6. Under **Select one of the following point-of-entry-options** select the second option, **Forward unencrypted traffic to Elastic Runtime Router**.
 7. 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.

9. 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.

- Configure your load balancer to append the `X-Forwarded-For` and `X-Forwarded-Proto` headers to client requests.

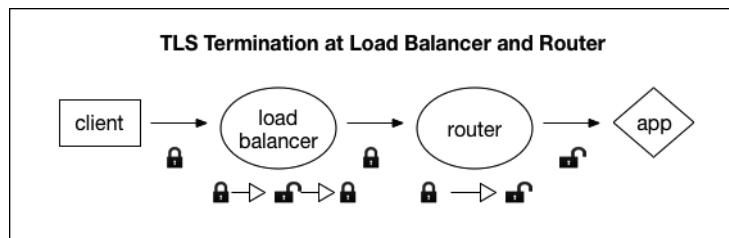
If the load balancer cannot be configured to provide the `X-Forwarded-For` header, the Gorouter will append it in requests forwarded to applications and system components, set to the IP address of the load balancer.

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 SSL/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 it does allow for use of multiple certificates under certain circumstances.

Certificate Guidelines

In this deployment scenario, the following guidelines apply:

- Certificates for the Elastic Runtime domains must be stored on the load balancer.
- On the Gorouter you can use a SAN certificate for all domains, or a standard certificate for a single domain that the load balancer has been configured to trust. In the latter case, however, you must disable hostname validation on the load balancer, as the domain of the certificate served by the Gorouter will not match requests sent to applications on Elastic Runtime.
- If you are concerned about hosting a certificate key on the Gorouter, you can generate certificates for your load balancer for the same domains with a different key. 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.

About Hostname Verification

Hostname verification between the load balancer and 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 SSL/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 requests it sends to the Gorouter.

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 Elastic Runtime 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
<code>*.cf</code>	A	<code>198.51.100.1</code>	<code>example.com</code>

2. Navigate to the Ops Manager Installation Dashboard.
3. Click the **Elastic Runtime** tile in the Installation Dashboard.
4. Click **Networking**.
5. For PCF deployments on OpenStack or vSphere, enter the load balancer IP address that you used to set up a wildcard DNS record in the **Router IPs** field. For more information, see the Elastic Runtime networking configuration topic for [OpenStack](#) or [vSphere](#).
6. Under **Select one of the following point-of-entry options** select the first option, **Forward SSL to Elastic Runtime Router**.
7. Enter your PEM-encoded certificate and your PEM-encoded private key in the fields under **Router SSL Termination Certificate and Private Key**. You can either upload your own certificate or generate a RSA certificate in Elastic Runtime. For options and instructions on creating a certificate for your wildcard domains, see the [Creating a Wildcard Certificate for PCF Deployments](#) topic.
8. If you want to use a specific set of SSL ciphers for the Router, configure **Router SSL Ciphers**. Enter a colon-separated list of custom SSL ciphers to pass to the router. Otherwise, leave this field blank.
9. 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.

10. Click **Save**.
11. 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.
12. 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.

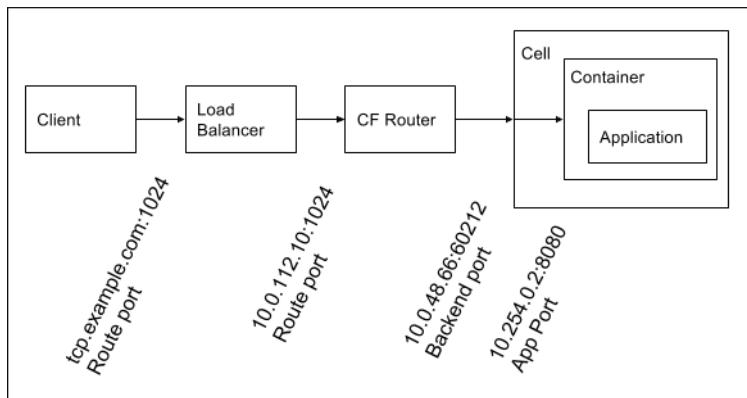
Enabling TCP Routing

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 Elastic Runtime](#) topic for your IaaS to configure your IP addresses for your

PCF deployment: [Amazon Web Services](#), [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 Elastic Runtime* topic for your IaaS to configure your ports for your PCF deployment: [Amazon Web Services](#), [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"}'
```

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. Obtain the secret for your `tcp_emitter` OAuth client from your manifest:

```
tcp_emitter:
  authorities: routing.routes.write,routing.routes.read
  authorized-grant-types: client_credentials,refresh_token
  secret: sample-secret
```

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.

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 cf Command Line Interface (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 [Modify Your Release Manifest](#) 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.

Modify Your Release Manifest

By default, the CF release manifest assigns port 4443 for TCP/WebSocket communications. If you have configured your load balancer to use a port other than 4443 for TCP/WebSocket traffic, you must edit your CF manifest to set the value of properties.logger_endpoint.port to the correct port. Locate the following section of your CF manifest and replace YOUR-WEBSOCKET-PORT with the appropriate value:

```
properties:  
logger_endpoint:  
port: YOUR-WEBSOCKET-PORT
```

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 Healthcheck Endpoints for CF Routers

Configure your load balancer to use the following HTTP healthcheck endpoints. Add the IP addresses of all router instances along with their corresponding port and path.

- HTTP Router (Gorouter): `http://GOROUTER_IP:8080/health`
- TCP Router: `http://TCP_ROUTER_IP:80/health`

The configuration above assumes the default healthcheck ports for the CF routers. To modify these ports, see the sections below.

Set the Gorouter Healthcheck Port

You can set the healthcheck port for the Gorouter in the cf-release manifest using the `router.status.port` property. The value of this property defaults to `8080`.

Set the TCP Router Healthcheck Port

You can set the healthcheck port for the TCP Router in the routing-release manifest using the `tcp_router.health_check_port` property. The value of this property defaults to `80`.

 **Note:** This property does not affect the healthcheck of the HAProxy deployed with cf-release.

Add a Healthcheck Endpoint for HAProxy

Configure your load balancer to use the following HTTP healthcheck endpoint: `http://HAProxy_IP:8080/health`.

The HAProxy included in cf-release is a legacy, optional component. Formerly, HAProxy handled TLS termination of HTTP requests, but Gorouter can now perform this termination. To make HAProxy highly available requires another load balancer in front of it, defeating the purpose. In environments where high availability is not required, DNS can resolve CF domains directly to single instances of the CF routers.

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.

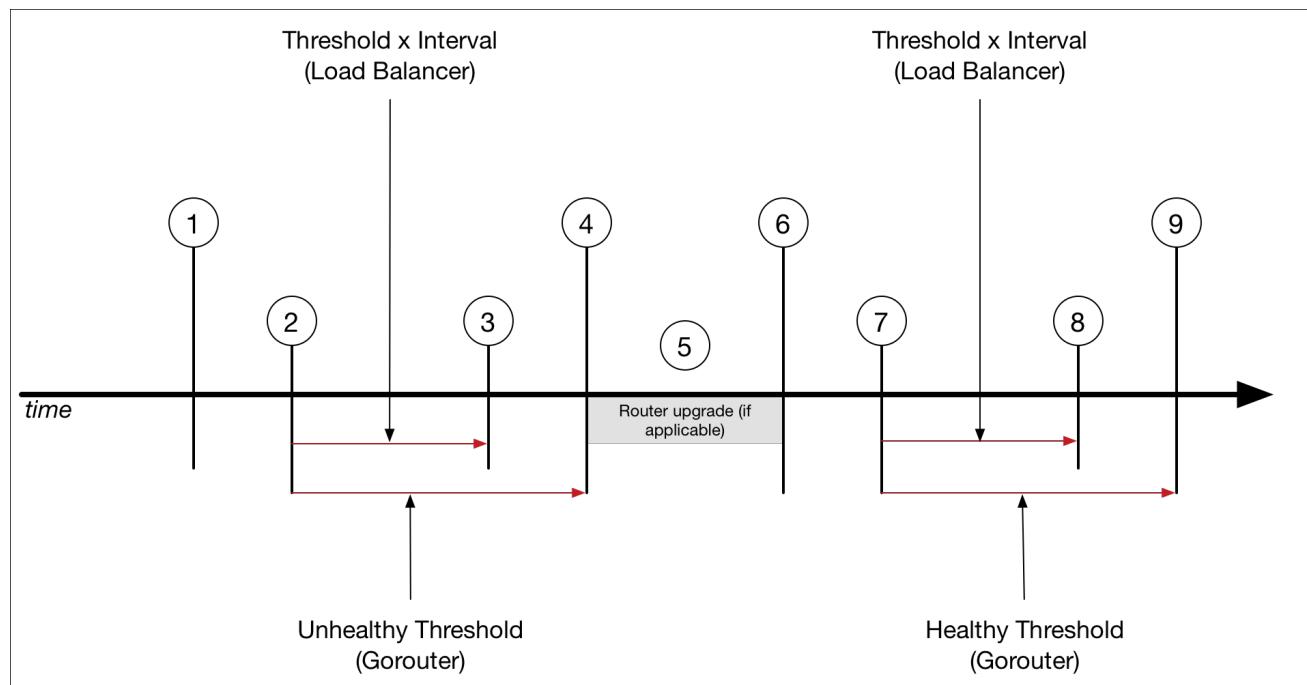
Elastic Runtime uses the following properties:

- **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: <ul style="list-style-type: none"> 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.

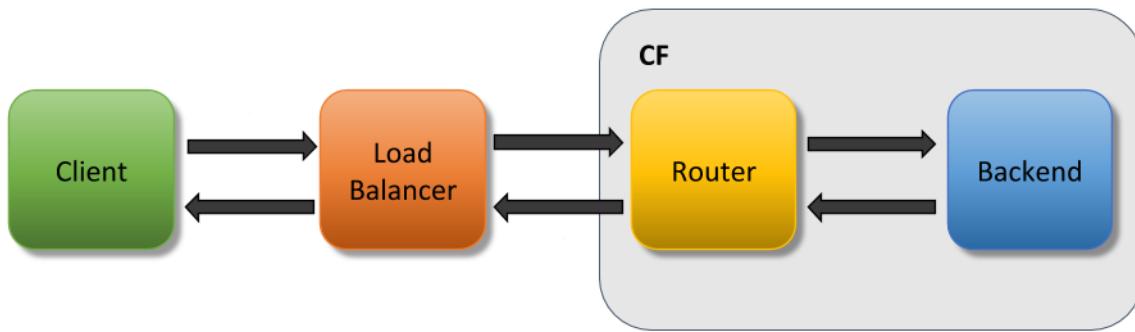
Troubleshooting Slow Requests in Cloud Foundry

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

To measure the total round-trip time for deployed app `app1`, run `time curl -v APP_ENDPOINT`:

```
$ 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. Run `cf logs APP_NAME`. (To see running app names, run `cf apps`.)
2. From another terminal pane, send a request to your app.
3. After your app returns a response, Ctrl-C to stop streaming `cf logs`:

```
$ 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-49ac3e286fe9"
^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-49ac3e286fe9"
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-49ac3e286fe9"
```

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 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 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

For a complete list of metrics available, see [Routing Metrics](#).

Router Idle Keepalive Connections

Page last updated:

This topic describes how to enable support for idle keepalive connections in the CF router, and considerations for configuration.

Support for keepalive connections is described in [Idle Keepalive Connections](#).

Enabling Idle Keepalive Connections for Gorouter to Backends

By default, support for keepalive connections 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 Elastic Runtime 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. Accomodating for growth of 2x, an operator might configure a maximum of 2000 idle connections per router.

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.

Using Windows Cells (BETA)

 **Note:** The procedures in this documentation are deprecated in Pivotal Cloud Foundry (PCF) 1.9.

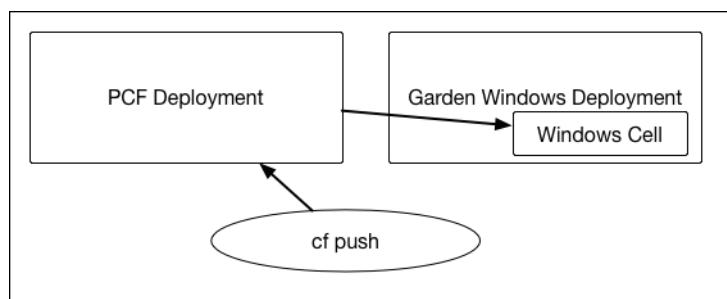
This documentation describes how operators install and manage Windows [cells](#) in PCF, and how developers push .NET apps to Windows cells.

 **Note:** You must have PCF 1.8 or later deployed on Amazon Web Services (AWS) or vSphere to install Windows cells.

Overview

Operators who want to run Windows cells in PCF in order to enable developers to push .NET apps can deploy the BOSH Release for Windows. The BOSH Release for Windows creates a separate BOSH deployment populated with the Garden Windows release, which runs on a Windows cell built from a Windows stemcell.

Once the Windows cell is running, developers can specify a Windows stack when pushing .NET apps from the command line. PCF passes the app to the Windows cell in the Garden Windows BOSH deployment. The diagram below illustrates the process.



Understanding BOSH Windows

- [Understanding Windows Cells](#)
- [Understanding Stemcell Security](#)

Installing Windows Cells

- [Building a Windows Stemcell](#)

 **Note:** The procedures in Building a Windows Stemcell are only required for vSphere deployments. If you are using Amazon Web Services (AWS), skip this topic and proceed directly to [Deploying the BOSH Release for Windows](#).
- [Deploying the BOSH Release for Windows](#)

Managing Windows Cells

- [Upgrading Windows Cells](#)
- [Rotating Credentials in Garden Windows](#)
- [Troubleshooting Windows Cells](#)

Developing on Windows Cells

- [Deploying .NET Apps to Windows Cells](#)

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

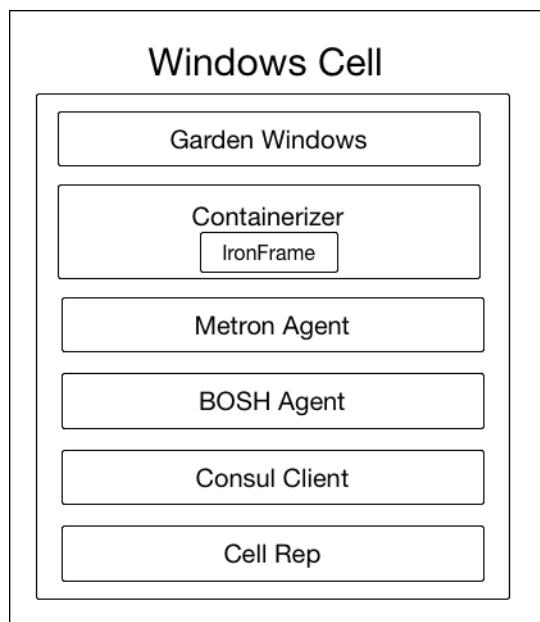
App instances in PCF run inside containers. [Garden](#) is the API that creates and manages these containers, and Garden Windows implements Garden on Windows.

By deploying the [BOSH Release for Windows](#), 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.

Understanding Stemcell Security

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, Vcap
Deny log on as a service	Guests, Vcap
Deny log on through Remote Desktop Services	Guests

Building a Windows Stemcell

This topic describes how to create a local [Concourse](#) pipeline to build a Windows stemcell for use with the BOSH Release for Windows. If you are deploying the BOSH Release for Windows on vSphere, you must follow the procedures in this topic to build a stemcell before performing the steps in [Deploying BOSH Windows](#).

 **Note:** The procedures in this topic are only required for vSphere deployments. If you are using Amazon Web Services (AWS), skip this topic and proceed directly to [Deploying the BOSH Release for Windows](#).

Requirements

Concourse is a pipeline-based continuous integration system. To learn more about Concourse, see the [documentation](#).

You must have the following to create a Concourse pipeline for building a Windows stemcell:

- A vSphere environment with outbound Internet and root access to an ESXi, which can be nested
- Access to an S3-compatible blobstore, either through Amazon Web Services (AWS) or a local [Minio](#) server
- The BOSH Release for Windows, downloaded from [Pivotal Network](#)

Step One: Prepare Your ESXi

1. SSH into the ESXi as root, replacing YOUR-ESXI with the IP address or domain name of the ESXi host. When prompted, enter your root password.

```
$ ssh root@YOUR-ESXI  
Password:
```

2. Run the following commands from the ESXi:

```
[root@:] esxcli system settings advanced set -o /Net/GuestIPHack -i 1  
[root@:] esxcli network firewall ruleset set --ruleset-id=gdbserver --enabled=yes
```

Step Two: Prepare Your Blobstore

You can use [AWS](#) or a local [Minio](#) server for your S3-compatible blobstore.

AWS

Perform the following steps to prepare and populate an AWS S3 blobstore:

1. Navigate to the AWS Console.
2. Click **S3** to open the S3 Management Console.
3. Click **Create Bucket**.
4. Enter a globally unique name for your S3 bucket under **Bucket Name**.
5. Under **Region**, select **US Standard**.
6. Click **Create**.
7. Select the newly created S3 bucket and click **Permissions**. Ensure that your user has **Upload/Delete** permissions. If necessary, navigate to the Identity and Access Management (IAM) console to ensure your user has the correct IAM permissions.

8. Download the following files:
 - The VMWare OVF Tool Linux 64-bit from [VMWare](#)
 - The UltraDefrag tool from [SourceForge](#)
 - The PSWindowsUpdate PowerShell module from [Microsoft](#)
 - The Windows Server 2012 R2 ISO from [Microsoft](#)

9. Rename the PSWindowsUpdate PowerShell module from `PSWindowsUpdate.zip` to `PSWindowsUpdate_v1.zip`:

```
$ mv PSWindowsUpdate.zip PSWindowsUpdate_v1.zip
```

10. Upload the VMWare OVF Tool `.bundle` file and the PSWindowsUpdate PowerShell module `.zip` file to your S3 bucket.

Minio

Perform the following steps to prepare and populate a Minio blobstore:

1. Install [Golang](#).
2. Install Minio by building it from source:

```
$ go get -u github.com/minio/minio
```

3. [Download](#) and install `mc`, the Minio Client.

4. Follow the instructions for your operating system in the Minio [readme](#) to run a local Minio server. When the server starts, Minio outputs the `AccessKey` and `SecretKey`. Record these values.

 **Note:** You must have at least 5% of your local disk free to run a local Minio server.

5. With your local Minio server running, add your server to your list of hosts:

```
$ mc config host add myminio http://10.0.0.10:9000 YOUR-ACCESS-KEY YOUR-SECRET-KEY
```

6. Create a bucket:

```
$ mc mb myminio/bosh-windows-bucket
```

7. Download the following files:
 - The VMWare OVF Tool Linux 64-bit from [VMWare](#)
 - The UltraDefrag tool from [SourceForge](#)
 - The PSWindowsUpdate PowerShell module from [Microsoft](#)
 - The Windows Server 2012 R2 ISO from [Microsoft](#)

8. Rename the PSWindowsUpdate PowerShell module from `PSWindowsUpdate.zip` to `PSWindowsUpdate_v1.zip`:

```
$ mv PSWindowsUpdate.zip PSWindowsUpdate_v1.zip
```

9. Copy the VMWare OVF Tool `.bundle` file and the PSWindowsUpdate PowerShell module `.zip` file into your Minio bucket:

```
$ mc cp ~/Downloads/VMware-ovftool-4.1.0-2459827-lin.x86_64.bundle myminio/bosh-windows-bucket
$ mc cp ~/Downloads/PSWindowsUpdate_v1.zip myminio/bosh-windows-bucket
```

Step Three: Prepare Your Pipeline

1. Change into the directory where you downloaded the BOSH Release for Windows files from Pivotal Network:

```
$ cd ~/bosh-windows
```

2. Open the `consumer-vars.yml` file and replace the `REPLACE_ME` placeholders as follows:

- `AWS_ACCESS_KEY` : Enter your AWS or Minio access key.
- `AWS_SECRET_KEY` : Enter your AWS or Minio secret key.
- `S3_OVFTOOL_BUCKET`, `S3_ULTRADEFRAG_BUCKET`, `S3_WINDOWS_UPDATE_BUCKET`, `S3_STEMCELL_OUTPUT_BUCKET`, `S3_WINDOWS_ISO_BUCKET` : Enter the name of the bucket you created in your S3-compatible blobstore. If you followed the steps above to set up a Minio server, the name is `bosh-windows-bucket`.
- `ESXI_REMOTE_ADDRESS` : Enter the IP address of your ESXi host.
- `ESXI_REMOTE_PASSWORD` : Enter the root password of your ESXi host.
- `ESXI_DATASTORE` : Enter the UUID of the ESXi datastore to store the VM.
- `ESXI_CACHE_DATASTORE` : Enter the UUID of the ESXi datastore to store the cache file for Packer. This can be the same value as `ESXI_DATASTORE`.
- `WINDOWS_PASSWORD` : Choose an Administrator password for the outputted stemcell. This is the password you use to log in to the Windows cell through a Remote Desktop Protocol (RDP) client. If the password does not meet the normal Windows complexity rules, the password will remain the default, `vagrant`.

Step Four: Deploy Concourse

 **Note:** When deploying Concourse locally, your local machine must have Internet access.

1. Download and install [VirtualBox](#).
2. Download and install [Vagrant](#).
3. Perform the following commands to spin up a local Concourse server:

```
$ vagrant init concourse/lite  
$ vagrant up
```

View the Concourse interface by navigating to <http://192.168.100.4:8080>.

4. Download the binary of `fly`, the Concourse CLI, from [Concourse](#), and place it in your `$PATH`.
5. Log in to your Concourse server:

```
$ fly -t local login -c http://192.168.100.4:8080
```

6. Set your pipeline:

```
$ fly -t local set-pipeline -p bosh-windows-consumer -c bosh-windows-consumer.yml -l consumer-vars.yml
```

7. If Concourse tells you the pipeline is paused, unpause it:

```
$ fly unpause-pipeline -t local -p bosh-windows-consumer
```

8. Navigate to your Concourse pipeline in a browser at <http://192.168.100.4:8080/teams/main/pipelines/bosh-windows-consumer>.
9. Kick off a build by selecting **create-stemcell** and clicking the plus sign in the upper right. After several hours, the outputted stemcell appears in the bucket you created in your S3-compatible blobstore.

Deploying BOSH Release for Windows

This topic describes how to deploy the BOSH Release for Windows to install Windows cells on your Pivotal Cloud Foundry (PCF) deployment.

 **Note:** The BOSH Release for Windows is currently in beta.

Requirements

To deploy the BOSH Release for Windows, you must have PCF 1.8 or later deployed to vSphere or Amazon Web Services (AWS).

- If your PCF deployment runs on vSphere, you must build your own stemcell by following the steps in the [Building a Windows Stemcell](#) topic before performing the procedures below.
- If your PCF deployment runs on AWS, you can use the stemcell included in the BOSH Release for Windows, but your deployment must be in `us-east-1`, `us-west-2`, or `eu-west-1`.

 **Note:** Once your Windows cell is running, you must disable FIPS as a Group Policy setting. If you fail to disable FIPS as a Group Policy setting, Garden Windows will not work.

Step 1: Prepare to Deploy

1. Ensure that you created a service network during your Ops Manager installation. A service network specifies a CIDR range within which Ops Manager does not provision VMs. You create a service network by selecting a checkbox in the **Create Networks** section of Ops Manager. See your IaaS-specific topic for configuring Ops Manager from the [Installing Pivotal Cloud Foundry](#) topic for more information.
2. Download all of the BOSH Release for Windows files from [Pivotal Network](#) to a single directory.
3. Prepare to SCP onto your Ops Manager VM.
 - For AWS, perform the following steps:
 1. In the EC2 instances page of your AWS Console, locate the FQDN of the Ops Manager VM.
 2. Locate the `ops_mgr.pem` private key file you used when installing Ops Manager, and ensure that you have added it to your list of private keys with the following terminal command:

```
$ ssh-add ops_mgr.pem
```

- For vSphere, perform the following steps:
 1. In vCenter, locate the FQDN of the Ops Manager VM.
 2. Locate the credentials you used to import the PCF `.ova` or `.ovf` file into your virtualization system.

Step 2: Deploy BOSH Release for Windows

1. In a terminal window, navigate to the directory where you downloaded the BOSH Release for Windows files. For example, if you downloaded the files to the `~/bosh-windows` directory, run the following command:

```
$ cd ~/bosh-windows
```

2. Securely copy `garden-windows-0.x.tgz`, `generate_manifest.rb`, and your stemcell file to your Ops Manager VM as `ubuntu@OPS-MANAGER-FQDN`.

 **Note:** For AWS, use the stemcell included in the BOSH Release for Windows. For vSphere, use the stemcell you built in the [Building a Windows Stemcell](#) topic.

The following example securely copies an AWS stemcell:

```
$ scp garden-windows-0.0.6.tgz generate_manifest.rb  
light-bosh-stemcell-0.0.46-aws-xen-hvm-windows2012R2-go_agent.tgz  
ubuntu@pcf.example.com:~
```

For vSphere, enter the password that you set during the `.ova` deployment into vCenter when prompted.

3. Follow the steps in the [Log into BOSH](#) section of the *Advanced Troubleshooting with the BOSH CLI* topic to target and log in to the BOSH Director. The steps vary slightly depending on whether your PCF deployment uses internal authentication or an external user store.
4. After you successfully log in to the BOSH Director, use the `bosh upload stemcell YOUR-WINDOWS-STEMCELL.tgz` command to upload the Windows stemcell to BOSH. Replace `YOUR-WINDOWS-STEMCELL.tgz` with the name of your Windows stemcell.

```
$ bosh upload stemcell YOUR-WINDOWS-STEMCELL.tgz
```

 **Note:** For AWS, your deployment must be in `us-east-1`, `us-west-2`, or `eu-west-1` to upload the stemcell to BOSH successfully.

5. Use the `bosh download manifest YOUR-PCF-DEPLOYMENT YOUR-PCF-MANIFEST.yml` command to download the manifest of your PCF deployment. Replace `YOUR-PCF-DEPLOYMENT` with the name of your PCF deployment, and `YOUR-PCF-MANIFEST.yml` with a manifest name to use to later steps.

```
$ bosh download manifest YOUR-PCF-DEPLOYMENT YOUR-PCF-MANIFEST.yml
```

 **Note:** You must know the name of your PCF deployment to download the manifest. To retrieve it, run `bosh deployments` to list your deployments and locate the name of your PCF deployment.

6. Use the manifest generation script included in the BOSH Release for Windows to generate a manifest for your deployment. You must specify either `vsphere` or `aws` depending on your IaaS. The following example uses AWS:

```
$ ./generate-manifest YOUR-PCF-MANIFEST.yml aws > garden-windows.yml
```

7. In a text editor, modify the generated manifest to replace the network name with the name of [your service network](#).

8. Upload the Garden Windows release to BOSH:

```
$ bosh upload release garden-windows-y-x.tgz
```

9. Deploy Garden Windows:

```
$ bosh -d garden-windows.yml deploy
```

Upgrading Windows Cells

This topic describes how to upgrade the Windows cells on a Pivotal Cloud Foundry (PCF) deployment. Operators may want to upgrade their Windows cells in response to the following events:

- For deployments that run on Amazon Web Services (AWS), the release of a new Windows stemcell included in the BOSH Release for Windows on [Pivotal Network](#).
- For deployments that run on vSphere, an update to the Windows Server 2012 operating system
- A new release of Garden Windows

Update the Stemcell

Perform the following steps to update the stemcell for your Windows cells:

1. Retrieve the new stemcell by performing the steps appropriate for your IaaS:
 - If your deployment runs on AWS, download the new stemcell from the latest BOSH Release for Windows on [Pivotal Network](#).
 - If your deployment runs on vSphere, ensure that you have followed the procedures in the [Building a Windows Stemcell](#) topic to create a Concourse pipeline for building stemcells. Navigate to your Concourse pipeline in a browser and kick off a build by selecting **create-stemcell** and clicking the plus sign in the upper right. Concourse uses the newest version of the Windows Server 2012 operating system to build a new stemcell. After several hours, the stemcell appears in the bucket you created in your S3-compatible blobstore.
2. Follow the steps in the [Log into BOSH](#) section of the Advanced Troubleshooting with the BOSH CLI topic to target and log in to your BOSH Director. The steps vary slightly depending on whether your PCF deployment uses internal authentication or an external user store.
3. After you have successfully logged in to your BOSH Director, upload the new stemcell to BOSH:

```
$ bosh upload stemcell YOUR-WINDOWS-STEMCELL.tgz
```

4. If necessary, download the manifest of your `garden-windows` deployment:

```
$ bosh download manifest garden-windows garden-windows.yml
```

5. Set your deployment to `garden-windows`:

```
$ bosh deployment garden-windows.yml
```

6. Edit your manifest:

```
$ bosh edit deployment
```

7. Locate the top-level property `stemcells` and update the version number of your Windows stemcell to the new version:

```
stemcells:  
- os: windows2012R2  
  alias: windows  
  version: 0.0.183
```

8. Save and exit the manifest.

9. Redeploy:

```
$ bosh -d garden-windows.yml deploy
```

Update the Garden Windows Release

Perform the following steps to update the Garden Windows release:

1. Download the new Garden Windows release tarball from the latest BOSH Release for Windows on [Pivotal Network](#).
2. Follow the steps in the [Log into BOSH](#) section of the Advanced Troubleshooting with the BOSH CLI topic to target and log in to your BOSH Director. The steps vary slightly depending on whether your PCF deployment uses internal authentication or an external user store.
3. If necessary, download the manifest of your `garden-windows` deployment:

```
$ bosh download manifest garden-windows garden-windows.yml
```

4. Set your deployment to `garden-windows`:

```
$ bosh deployment garden-windows.yml
```

5. Upload the new Garden Windows release to BOSH:

```
$ bosh upload release garden-windows-y-x.tgz
```

6. Edit your manifest:

```
$ bosh edit deployment
```

7. Locate the top-level property `releases` and update the version number of the Garden Windows release to the new version:

```
releases:  
- name: garden-windows  
  version: 0.0.6
```

8. Save and exit the manifest.

9. Redeploy:

```
$ bosh -d garden-windows.yml deploy
```

Rotating Credentials in Garden Windows

This topic describes how to rotate the credentials for your Garden Windows release.

When operators rotate credentials for a Pivotal Cloud Foundry (PCF) deployment, this rotation does not automatically take effect in their Garden Windows release. To ensure that the Garden Windows release shares the new credentials of the PCF deployment, they must regenerate the manifest for the Garden Windows release and redeploy it.

Perform the following steps to rotate your Garden Windows credentials:

1. Follow the steps in the [Log into BOSH](#) section of the Advanced Troubleshooting with the BOSH CLI topic to target and log in to your BOSH Director. The steps vary slightly depending on whether your PCF deployment uses internal authentication or an external user store.
2. Download the manifest of your PCF deployment:

```
$ bosh download manifest YOUR-PCF-DEPLOYMENT YOUR-PCF-MANIFEST.yml
```

 **Note:** You must know the name of your PCF deployment to download the manifest. To retrieve it, run `bosh deployments` to list your deployments and locate the name of your PCF deployment.

3. Use the manifest generation script included in the BOSH Release for Windows on [Pivotal Network](#) to regenerate the manifest for Garden Windows. You must specify either `vsphere` or `aws` depending on your IaaS. The following example uses AWS:

```
$ ./generate-manifest YOUR-PCF-MANIFEST.yml aws > garden-windows.yml
```

4. Upload the Garden Windows release to BOSH:

```
$ bosh upload release garden-windows-y-x.tgz
```

5. Redeploy Garden Windows:

```
$ bosh -d garden-windows.yml deploy
```

Troubleshooting Windows Cells

This topic describes how to troubleshoot Windows cells on a Pivotal Cloud Foundry (PCF) deployment.

To perform the troubleshooting procedures in this topic, you must first log in to BOSH and set your current deployment to Garden Windows by performing the following steps:

1. Follow the steps in the [Log into BOSH](#) section of the Advanced Troubleshooting with the BOSH CLI topic to target and log in to your BOSH Director. The steps vary slightly depending on whether your PCF deployment uses internal authentication or an external user store.
2. If necessary, download the manifest of your Garden Windows deployment:

```
$ bosh download manifest garden-windows garden-windows.yml
```

3. Set your deployment to `[garden-windows]`:

```
$ bosh deployment garden-windows.yml
```

4. Continue to [retrieving logs](#) or [connecting](#) to your Windows cell.

Retrieve Logs

Perform the following steps to retrieve the logs for the Windows cell:

1. Download the logs, replacing `[YOUR-LOGS-DIR]` with your destination directory:

```
$ bosh logs cell_windows 0 --dir YOUR-LOGS-DIR
```

2. Your logs appear as a tarball in the destination directory you specified. Change into the directory and unzip the tarball:

```
$ tar xvf cell_windows.0.2016-10-04-15-52-37.tgz
```

3. Examine the logs. Each component on the cell has its own logs directory:

- `/consul_agent_windows/`
- `/garden-windows/`
- `/metron_agent_windows/`
- `/rep_windows/`

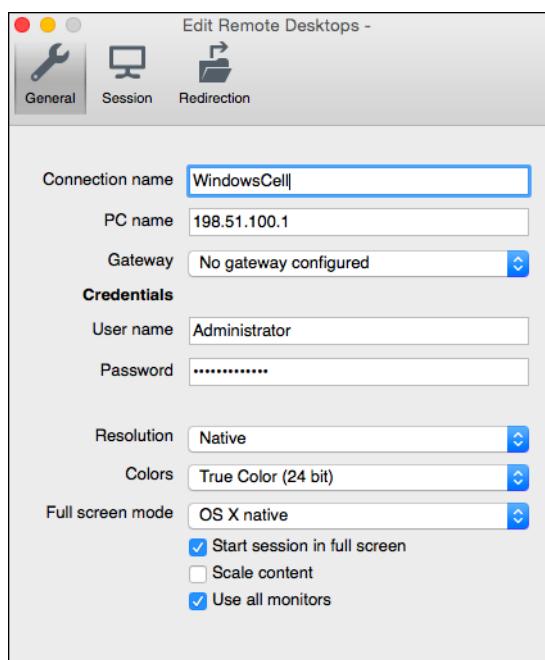
Connect to the Windows Cell

Perform the following steps to connect to your Windows cell to run diagnostics:

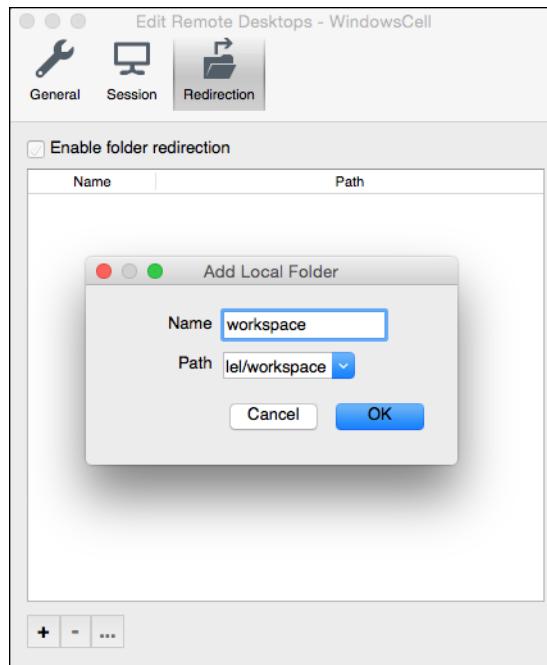
1. Download and install a Remote Desktop Protocol (RDP) client.
 - For Mac OS X, download the Microsoft Remote Desktop app from the [Mac App Store](#).
 - For Windows, download the Microsoft Remote Desktop app from [Microsoft](#).
 - For Linux/UNIX, download a RDP client like [rdesktop](#).
2. Retrieve the IP address of your Windows cell:

```
$ bosh vms garden-windows
Acting as user 'director' on deployment 'garden-windows' on 'p-bosh-1170e9b438cb29ff7c63'
Director task 274
Task 274 done
+-----+-----+-----+-----+
| VM | State | AZ | VM Type | IPs |
+-----+-----+-----+-----+
| cell_windows/0 (03e221b3-3222-5e1e-eedd-b92221ff88e1) | running | default | xlarge | 198.51.100.1 |
+-----+-----+-----+-----+
VMs total: 1
```

3. Retrieve the Administrator password for your Windows cell by following the steps for your IaaS:
 - On vSphere, this is the value of `WINDOWS_PASSWORD` in the `consumer-vars.yml` file you used to build a stemcell in the [Building a Windows Stemcell](#) topic.
 - On Amazon Web Services (AWS), navigate to the AWS EC2 console. Right-click on your Windows cell and select **Get Windows Password** from the drop-down menu. Provide the local path to the `ops_mgr.pem` private key file you used when installing Ops Manager and click **Decrypt password** to obtain the Administrator password for your Windows cell.
4. Open your RDP client. The examples below use the Microsoft Remote Desktop app.



5. 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 `Administrator`.
 - **Password:** Enter the password of your Windows cell that you obtained above.
6. 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**.
7. 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.
8. In the RDP session, you can use the following tools to diagnose problems with your Windows cell:
- [Hakim](#)
 - [Consul CLI](#)

Hakim

Hakim is a diagnostic tool that reveals common configuration issues with Windows cells. Perform the following steps to use Hakim:

1. The Hakim binary is included in the [DiegoWindows](#) zip file in the Pivotal Cloud Foundry Elastic Runtime product on [Pivotal Network](#). You can place the Hakim binary on your Windows cell in one of two ways:
 - Download the [DiegoWindows](#) zip file, unzip it, and place `hakim.exe` into a local directory that you mount as a drive on the Windows cell by following the [steps](#) above.
 - In the RDP session, open Internet Explorer and log in to [Pivotal Network](#) to download the [DiegoWindows](#) zip file directly to your Windows cell.
2. Open a PowerShell window and change into the directory that contains `hakim.exe`:

```
PS C:\Users\Administrator> cd Downloads
```

3. Run `hakim.exe`:

```
PS C:\Users\Administrator\Downloads> .\hakim.exe
```

Hakim only outputs to the PowerShell if it detects errors. Refer to the [section](#) below for a list of Hakim error messages and their possible solutions.

Hakim Error Messages

Processes

The following processes are not running

This usually indicates a failed deployment. Try [redeploying](#) the BOSH Release for Windows.

NTP

There was an error detecting ntp synchronization on your machine. An accurate system clock is essential for internal Cloud Foundry metric reports. Please configure your NTP settings, if not already done. We recommend that your firewall have outbound rules set for UDP on port 123. In addition, ensure that your 'DnsCache' service is running

If NTP is not configured, clock skew with other PCF components can occur. Clock skew can result in odd errors, such as not receiving any metrics from apps running on the affected machine. Ensure that you are using the same NTP server on your Windows cell as the rest of your PCF deployment.

Firewall

Windows firewall service is not enabled. The Windows firewall is required in order to enforce Application Security Group rules. Running without the firewall is possible, but strongly not recommended.

Garden Windows enforces PCF security group settings for apps running on the Windows cell through the Windows firewall. Apps can run without this, but security groups do not work correctly and apps have unrestricted network access.

To resolve this error, enable the Windows firewall. Perform the following steps in your RDP session to access the Windows firewall configuration:

1. Open the Server Manager from the task bar.
2. Click **Tools** in the upper right and select **Windows Firewall with Advanced Security**.
3. Configure and enable the Windows firewall.

Fair Share

Fair Share CPU Scheduling must be disabled

You must disable Fair Share CPU scheduling for your Windows cell to function properly. Perform the following steps in your RDP session:

1. Open the Registry Editor at `C:\Windows\regedit.exe`.
2. Navigate to `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Quota System`.
3. Double-click the `EnableCpuQuota` value.
4. Change the **Value data** from `1` to `0`.
5. Click **OK**.

Container

Failed to create container

This usually indicates an issue with the Windows containerization service. Contact [Pivotal Support](#) and provide the full output of this error.

Consul

Failed to resolve consul host

This usually indicates interference with DNS resolution on your Windows cell. To resolve this error, perform the following steps in your RDP session to set `127.0.0.1` as the primary DNS server for the active network adapter:

1. Open the Control Panel.
2. Click **Network and Internet**
3. Click **Network and Sharing Center**.
4. Click **Change adapter settings** on the left.

5. Double-click your active network adapter.
6. Click **Properties**.
7. Select **Internet Protocol Version 4 (TCP/IPv4)**.
8. Click **Properties**.
9. Ensure that **Use the following DNS server addresses** is selected and enter `127.0.0.1` for **Preferred DNS server**.
10. Click **OK**.

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.

After operators have [deployed](#) the BOSH Release for Windows to install a Windows cell on their Pivotal Cloud Foundry (PCF) deployment, developers can push .NET apps to the Windows cell. Developers can push both [OWIN](#) and non-OWIN apps, and can push apps that are served by [Hostable Web Core](#) or [self-hosted](#).

Push a .NET App

By default, PCF serves .NET apps with Hostable Web Core, which 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. Target the Cloud Controller of your PCF deployment:

```
$ cf api api.YOUR-SYSTEM-DOMAIN
```

2. Push your .NET app, replacing `APP-NAME` with the name of your app:

```
$ cf push APP-NAME -s windows2012R2 -b binary_buildpack
```

Push your app from a directory containing either a `.exe` binary or a valid `Web.config` file for .NET apps. Alternatively, add the `-p` flag to your `cf push` command and specify the path to the directory that contains the `.exe` or `Web.config` file.

The push command must include the following flags:

- You specify the stack as `windows2012R2` to instruct PCF to run the app in the Windows cell.
- You specify the buildpack as `binary_buildpack` to expedite the staging process, because the binary buildpack is lightweight. Garden Windows does not use the binary buildpack to stage your app.

3. Wait for your app to stage and start. If an error occurs, see the [Troubleshoot App Errors](#) section.

Push a Self-Hosted App

Developers can choose to push a self-hosted app instead of using Hostable Web Core. Self-hosted apps combine the server code with the app code.

Perform the following steps to push a self-hosted app:

1. Target the Cloud Controller of your PCF deployment:

```
$ cf api api.YOUR-SYSTEM-DOMAIN
```

2. Push your .NET app from the app root, replacing `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 an error occurs, see the [Troubleshoot App Errors](#) section.

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. Push your service from the directory containing the published web service, replacing `SERVICE-NAME` with the name of your service:

```
$ cf push SERVICE-NAME -s windows2012R2 -b binary_buildpack -u none
```

The push command must include the following flags:

- You specify the stack as `windows2012R2` to instruct PCF to run the app in the Windows cell.
- You specify the buildpack as `binary_buildpack` to expedite the staging process, because the binary buildpack is lightweight. Garden Windows does not use the binary buildpack to stage your app.

The push command can include the following optional flags:

- If you are not pushing your service from the directory containing the published web service, add the `-p` flag to your `cf push` command and specify the path to the directory that contains the published web service.
- If you do not have a route serving `/`, add the `-u none` flag to disable the health check.

4. If the push command is successful, locate the portion of the 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: binary_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.

See 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](#) will route the request 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:

1. Modify the WSDL file by following the instructions in [Modify a Web Service's WSDL Using a SoapExtensionReflector](#) from the Microsoft Developers Network.
2. Instruct the developers of external applications that consume the web service to perform the steps in the [section](#) below.

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 [above](#).
4. In Microsoft Visual Studio, right-click on your application in the **Solution Explorer** and select **Add** and then **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. Operators should see the [Troubleshooting Windows Cells](#) topic for information about how to troubleshoot their 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 specifying 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](#)
- [Managing Orgs and Spaces Using Apps Manager](#)
- [Managing User Roles with Apps Manager](#)
- [Managing Apps and Service Instances Using Apps Manager](#)
- [Viewing ASGs in Apps Manager](#)
- [Monitoring Instance Usage with Apps Manager](#)
- [Configuring Spring Boot Actuator Endpoints for Apps Manager](#)
- [Using Spring Boot Actuator Endpoints with Apps Manager](#)

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, manage, and delete 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	<code>create-org</code>	†	†	†
Delete an org	<code>delete-org</code>	Yes	No	No
Rename an org	<code>rename-org</code>	Yes	No	No
View org members	<code>org-users</code>	Yes	Yes	Yes
Assign user a role in org	<code>set-org-role</code>	‡	‡	No
Remove org role from user	<code>unset-org-role</code>	‡	‡	No
View space members	<code>space-users</code>	Yes	Yes	Yes
Assign user a role in space	<code>set-space-role</code>	‡	‡	No
Remove space role from user	<code>unset-space-role</code>	‡	‡	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:

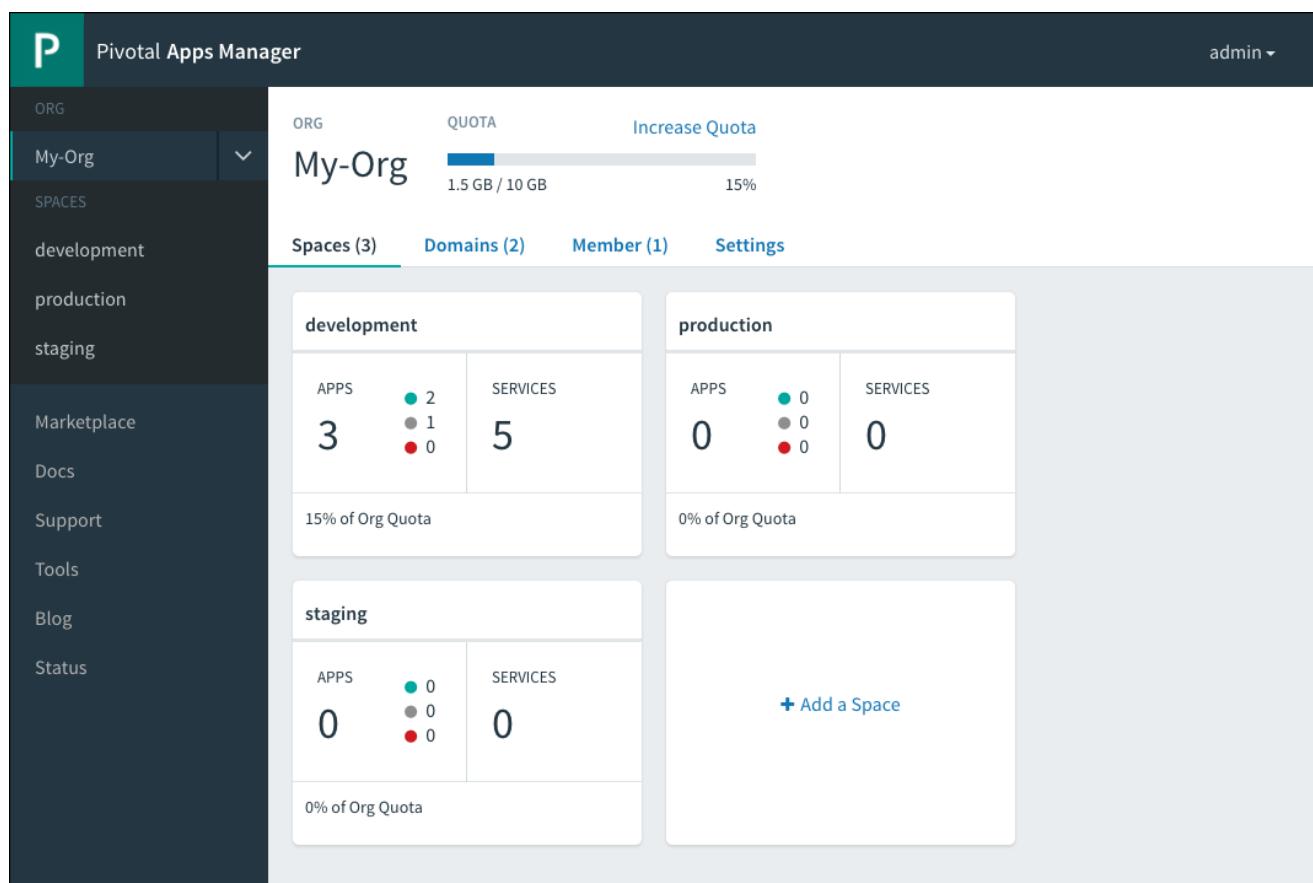
This topic discusses how to view and manage orgs and spaces in Apps Manager.

 **Note:** To manage a space, you must have Space Manager permissions in that space.

To perform the following steps, 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.



The screenshot shows the Pivotal Apps Manager interface. The top navigation bar includes the Pivotal logo, the title "Pivotal Apps Manager", and a user dropdown labeled "admin". The left sidebar lists various orgs: "My-Org" (selected), "development", "production", "staging", "Marketplace", "Docs", "Support", "Tools", "Blog", and "Status". The main content area is titled "My-Org" and shows a quota of "1.5 GB / 10 GB" (15%). Below this, there are three tabs: "Spaces (3)", "Domains (2)", and "Member (1)". The "Spaces (3)" tab is active, displaying three cards for "development", "production", and "staging". Each card shows the count of "APPS" and "SERVICES". For "development": APPS 3 (2 green, 1 grey, 0 red), SERVICES 5. For "production": APPS 0 (0 green, 0 grey, 0 red), SERVICES 0. For "staging": APPS 0 (0 green, 0 grey, 0 red), SERVICES 0. Below each card, it says "15% of Org Quota" and "0% of Org Quota" respectively. A button "+ Add a Space" is located at the bottom right of the space section.

Space	APPS	SERVICES
development	3 (2 green, 1 grey, 0 red)	5
production	0 (0 green, 0 grey, 0 red)	0
staging	0 (0 green, 0 grey, 0 red)	0

To view spaces in a different org, use the drop-down menu to change the org.

The screenshot shows the 'Org' section of the Pivotal Apps Manager interface. It lists several organizations: 'My-Org', 'system', 'My-Org', 'Second-Org', and 'Another-Org'. At the bottom, there is a blue button labeled '+ Create a New 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.

Manage a Space

The space page displays the apps and service instances associated with the selected space.

The screenshot shows the 'development' space page. The left sidebar shows 'My-Org' selected. The main area shows the 'development' space with 2 running apps: 'spring-music' and 'springpong'. The 'Apps' tab is selected, showing a table with columns: NAME, INSTANCES, MEMORY, LAST PUSH, and ROUTE. Each app row includes a link to its route.

NAME	INSTANCES	MEMORY	LAST PUSH	ROUTE
spring-music	1	512MB	5 months ago	https://spring-music-dissimilatory-pyrop...
springpong	1	512MB	5 months ago	https://springpong.a1-app.cf-app.com

The **Apps** tab shows the **Name**, the number of **Instances**, the amount of **Memory** available, the time since the **Last Push**, and the **Route** for each app.

Apps				
NAME	INSTANCES	MEMORY	LAST PUSH	ROUTE
● spring-music	1	512MB	15 minutes ...	http://spring-music-dissimilatory-pyr...
● springpong	1	512MB	12 minutes ...	http://springpong.a1-app.cf-app.com

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 Service
Service	Name	Bound Apps	Plan	
 MySQL for Pivotal Cloud Foundry	mysql-db	1	free - (MONTH)	>

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.
- Delete the space by clicking **Delete Space**.

SPACE development 0 Running
2 Stopped
0 Crashed

Apps (2) Services Settings

Space Name Update Cancel

Security Groups A collection of egress rules that specify one or more individual protocols, ports, and destinations. [Learn more](#)

> public_networks	running, staging
> dns	running, staging
> sshfs-service	running
> p-mysql	running
> southcentral	running

Delete Space Delete Space
This will permanently delete all of the apps in this 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.

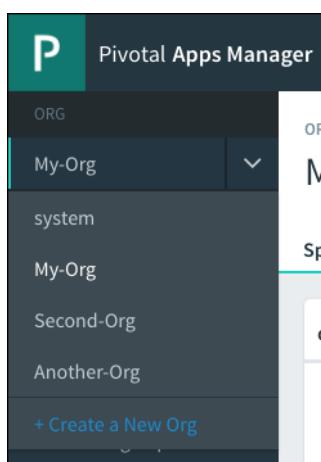
Admins, Org Managers, and Space Managers can assign user roles with Apps Manager or with the Cloud Foundry Command Line Interface (cf CLI). For more information, see the [Users and Roles](#) section of the *Getting Started with the cf CLI* topic.

Managing 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.

Spaces (3) **Domains (2)** **Members (6)** **Settings**

My-Org		Invite New Members
MEMBER ▾	ORG MANAGER	ORG AUDITOR
alice@example.c... <input checked="" type="checkbox"/>	<input type="checkbox"/>	
Remove User		
bob@example.com <input type="checkbox"/>	<input checked="" type="checkbox"/>	
Remove User		
claricep@example... <input type="checkbox"/>	<input checked="" type="checkbox"/>	
Remove User		
daviddavidson@... <input type="checkbox"/>	<input type="checkbox"/>	
Remove User		
eddieg@example... <input type="checkbox"/>	<input type="checkbox"/>	
Remove User		

ORG ROLES

ORG MANAGER
Can invite users and manage user roles in the org and all spaces

ORG AUDITOR
Can view members, roles, domains, and current org quota information

- 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.

Managing 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.

- In the **Members** tab of an org, click the drop-down menu to view spaces in the org.

- Use the drop-down menu to select a space.

- The **Members** panel displays all members of the org. Select a checkbox to grant an app space role to a user, or clear a checkbox to revoke a role from a user.

Spaces (3) **Domains (2)** **Members (6)** **Settings**

development				Invite New Members
MEMBER ▲	SPACE MANAGER	SPACE DEVELOPER	SPACE AUDITOR	
alice@example.com	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
bob@example.com	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
claricep@example...	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
daviddavidson@...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
eddieg@example...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
pspinrad@pivotal...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

SPACE ROLES

SPACE MANAGER
Can invite users and manage user roles in a given space

SPACE DEVELOPER
Can create, delete, and manage applications and services, and has full access to app logs and reports

SPACE AUDITOR
Has read-only access to space information, app logs, and reports

- **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.

Inviting New Users

1. On the Org dashboard, click the **Members** tab.

Spaces (3) **Domains (9)** **Member (1)** **Settings**

My-Org			Invite New Members
MEMBER ▲	ORG MANAGER	ORG AUDITOR	
admin	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

2. Click **Invite New Members**. The **Invite New Team Member(s)** form appears.

Spaces (3) **Domains (9)** **Member (1)** **Settings**

Invite New Team Member(s)

Add Email Addresses Use commas to separate emails

Assign Org Roles

ORG	ORG MANAGER	ORG AUDITOR
My-Org	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Select All		

ORG ROLES

ORG MANAGER
Can invite users and manage user roles in the org and all spaces

ORG AUDITOR
Can view members, roles, domains, and current org quota information.

Assign Space Roles

SPACE	SPACE MANAGER	SPACE DEVELOPER	SPACE AUDITOR
development	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

SPACE ROLES

SPACE MANAGER
Can invite users and manage user roles in a given space

SPACE DEVELOPER
Can create, delete, and manage roles in a given space

- 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.
- 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.
- Click **Send Invite**. The Apps Manager sends an email containing an invitation link to each email address that you specified.

Removing a User From a Space

To remove a user from a space, remove all user roles for the user in the space.

The user remains visible in the list unless you remove the user from the org.

Removing a User From an Org

- On the Org dashboard, click the **Members** tab.

[Spaces \(3\)](#) [Domains \(2\)](#) [Members \(6\)](#) [Settings](#)

My-Org		Invite New Members
MEMBER ▾	ORG MANAGER	ORG AUDITOR
alice@example.c... <input checked="" type="checkbox"/>	<input type="checkbox"/>	
Remove User		
bob@example.com <input type="checkbox"/>	<input checked="" type="checkbox"/>	
Remove User		
claricep@example... <input type="checkbox"/>	<input checked="" type="checkbox"/>	
Remove User		
daviddavidson@... <input type="checkbox"/>	<input type="checkbox"/>	
Remove User		
eddieg@example... <input type="checkbox"/>	<input type="checkbox"/>	
Remove User		

ORG ROLES

ORG MANAGER

Can invite users and manage user roles in the org and all spaces

ORG AUDITOR

Can view members, roles, domains, and current org quota information

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 the **Remove** button to confirm user account deletion from the org.

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, 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. This directs you to the app page, where 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.

The screenshot shows the Apps Manager interface for the 'My-app' application. At the top, there are three status indicators: a blue square (stopped), a blue circle with a white 'C' (restarting), and a green circle (running). To the right of these is a 'View App' button. Below the status indicators, there are tabs for Overview, Services, Route (1), Logs, Tasks, and Settings. The Buildpack is listed as ruby 1.6.34. On the left, there's a sidebar with 'Events' showing a 'Started app' entry from 02/28/2017 at 07:04:20 PM UTC and an 'Updated app' entry from the same date and time. The main content area is titled 'Scaling'. It has fields for 'Instances' (set to 1), 'Memory Limit' (1 GB), and 'Disk Limit' (1 GB). A 'Scale App' button is located at the top right of this section. Below the scaling section is a table titled 'Instances' with columns for #, CPU, MEMORY, DISK, and UPTIME. The data row shows 0 instances with 0% CPU, 0 Bytes memory, 0 Bytes disk, and a uptime of 4 hr 15 min.

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

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.

1. Under **Scaling**, adjust the number of **Instances**, the **Memory Limit**, and the **Disk Limit** as desired.
2. Click **Scale App**.

Bind or Unbind a Service

1. Click **Services**.
2. To bind your app to a service, click **Bind a Service**.

Bound Services

Bind a Service

select a service

Bind Cancel Go to the Marketplace

No services bound to this app. [Add a service](#) from the Marketplace.

3. To bind your app to an existing service instance, choose the service instance from the dropdown menu, and click **Bind**. To bind your app to a new service instance, click **Go to the Marketplace** to choose a service from the Marketplace.
4. 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**.

Routes

Map a Route

<http://spring-music-ageless-hydrazoate.cfapps.io>

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.

Logs

▶

```

2016-06-15T13:45:22.512-07:00 [API] [OUT] Updated app with guid e0cb1bb7-71c6-49e2-bb6a-
522991903d9f {"instances":>1}
2016-06-15T13:45:22.523-07:00 [CELL] [OUT] Exit status 0
2016-06-15T13:45:22.524-07:00 [APP] [OUT] [CONTAINER]
org.apache.coyote.http11.Http11NioProtocol INFO Pausing ProtocolHandler ["http-nio-8080"]
2016-06-15T13:45:22.578-07:00 [APP] [OUT] [CONTAINER]
org.apache.catalina.core.StandardService INFO Stopping service Catalina
2016-06-15T13:45:22.582-07:00 [APP] [OUT] [CONTAINER] lina.core.ContainerBase.[Catalina].
[localhost].[/] INFO Destroying Spring FrameworkServlet 'dispatcher'

```

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**.

APP
My-app ■ C ● Running

View App ▼

Overview Services (3) Routes (3) Logs Tasks Settings Buildpack: Ruby

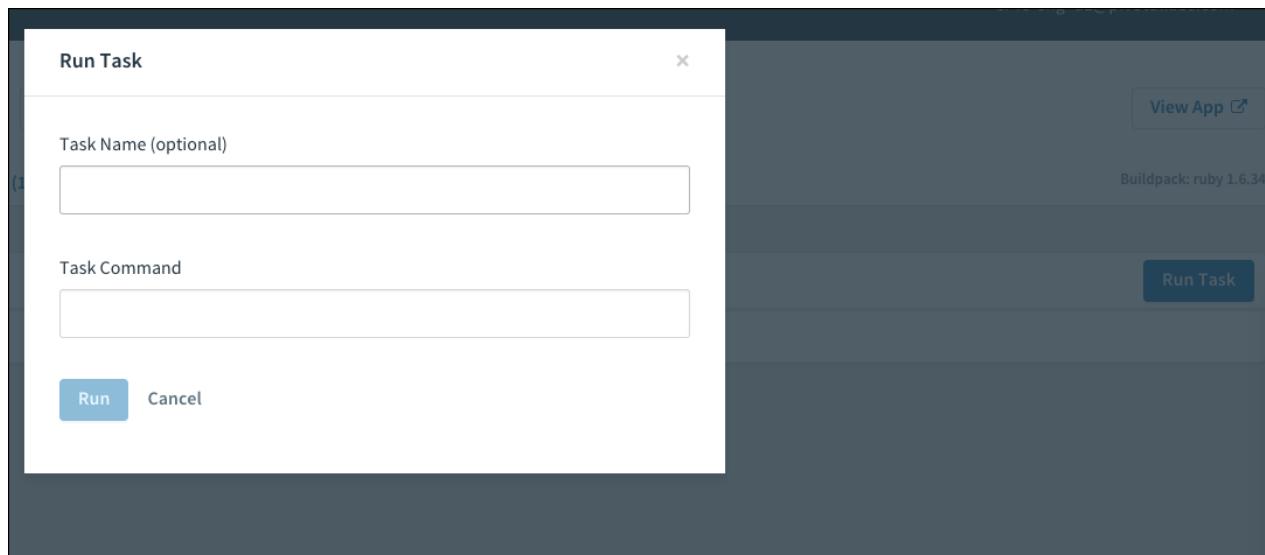
Tasks

TASK ID	State	Start Time	Task Name	Command
(1)				

Run a task

Run a Task

1. Click **Run a Task** to create a task.
2. (Optional) Enter a **Task Name**.
3. Enter the **Task Command**.



View Settings

Click the **Settings** tab. In this tab you can do the following:

- Rename the app.
- View or add Environment Variables associated with the app.
- View the Application Security Groups (ASGs) associated with the app.
- Delete the app.

APP

My-app   Running View App 

Overview Services (3) Routes (3) Logs Tasks **Settings** Buildpack: Ruby

App Name Update Cancel

Info Buildpack Git: [Ruby Buildpack](#)
Start Command: Set by buildpack
Stack: cflinuxfs2 (Cloud Foundry Linux-based file)

User Provided Environment Variables Reveal User Provided Env Vars

Environment Variables Reveal Env Vars
Defined by the runtime and buildpack. [Learn more](#)

Security Groups
A collection of egress rules that specify one or more individual protocols, ports, and destinations. [Learn More](#)

- > public_networks
- > dns
running, staging
- > ssh-logging
running, staging

Delete App Delete App
This will permanently delete the app and all of its data.

View or Add Environment Variables

Follow the procedure 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**.

User Provided Environment Variables

Name	Value

Save **Cancel** **+**

To view a user-provided environment variable, click **Reveal Env Vars.**

Environment Variables

Defined by the runtime and buildpack. [Learn more](#)

```
{  
  "staging_env_json": {},  
  "running_env_json": {},  
  "system_env_json": {  
    "VCAP_SERVICES": {}  
  },  
}
```

 **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 restart` command.

Manage a Service Instance

On the space page click **Services**, then click the service instance you want to manage. This directs you to the service instance page, where you can bind or unbind apps, 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 or Unbind an App

Follow the steps below to bind or unbind an app.

SERVICE

 MySQL for Pivotal Cloud Foundry [Manage](#) | [Docs](#) | [Support](#)

INSTANCE NAME	SERVICE PLAN
mysql-db	100mb

App Binding (1) **Plan** **Settings**

Bound Apps

springpong

Edit Bindings

1. Click **Edit Bindings**.
2. Select the checkbox for the apps you want to bind to or unbind from your service instance.
3. Click **Save**.

View or Change Your Service Plan

Follow the steps below to view or change your service plan.

1. Click **Plan**.

The screenshot shows the MySQL for Pivotal Cloud Foundry service plan page. At the top, it displays the service name "MySQL for Pivotal Cloud Foundry" and the instance name "mysql-db" with a "100mb" service plan. Below this, there are three tabs: "App Binding (1)", "Plan" (which is selected), and "Settings". On the left, a list of available plans is shown: "100mb free", "1gb free" (selected), "5gb free", and "20gb free". To the right of the selected plan, its details are listed: "1gb free", "Shared MySQL server", "1000 MB storage", and "40 concurrent connections". A blue button labeled "Select this plan" is visible at the bottom of this section.

2. Review your current plan information.
3. To change your plan, select a new plan from the list and click **Select this plan**.

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

Follow the steps below to rename or delete your service instance.

1. Click **Settings**.

SERVICE INSTANCE NAME SERVICE PLAN
The Fake Broker my-service fake-async-plan

[Overview](#) [Plan](#) [Settings](#)

Service Instance Name [Update](#) [Cancel](#)

Configure Instance
 For a list of supported configuration parameters, see documentation for the particular service offering.

<input type="text" value="Name"/>	<input type="text" value="Value"/>	+
-----------------------------------	------------------------------------	-------------------

[Update](#) [Cancel](#)

Delete Service Instance
 This will permanently delete the service and all of its data. [Delete Service Instance](#)

- 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.

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.

Overview Plan Settings Docs Support Manage

Bound Apps		Bind App
my-super-app		X
the-other-app		X

Bound Routes		Bind Route
my-super-app-url		X
the-other-app-url		X

Service Key Credentials		Create Service Key
external-app		X

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**.

Create Service Key

Service Key Name

Show Advanced Options Cancel Create

3. (Optional) Click **Show Advanced Options**. Under **Arbitrary Parameters**, enter any additional service-specific configuration in the **Name** and **Value** fields.

Create Service Key

Service Key Name

Arbitrary Parameters

Name	Value	+
------	-------	---

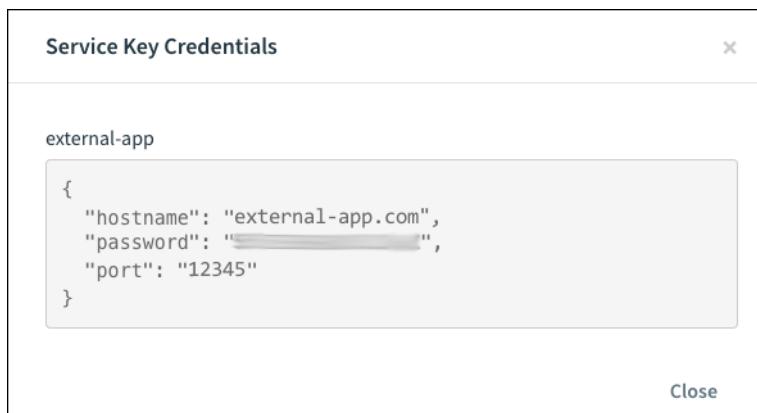
Hide Advanced Options Cancel Create

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.

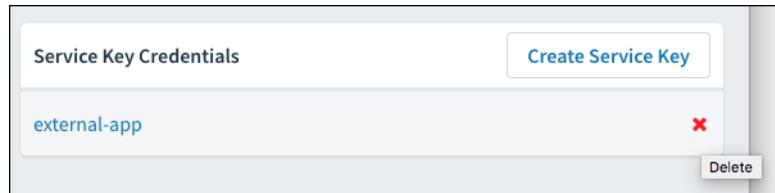
1. To view the credentials for a particular service instance, click the service instance name under **Service Key Credentials**. The JSON object containing the credentials appears.



2. Click **Close**.

Delete Service Key

To delete a service key, click the red X next to the service instance name.



Viewing ASGs in Apps Manager

Page last updated:

About ASGs

Application Security Groups (ASGs) are 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 on 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.

The screenshot shows the 'Security Groups' section of the Apps Manager. On the left, there's a sidebar with a 'Security Groups' heading and a brief description: 'A collection of egress rules that specify one or more individual protocols, ports, and destinations.' Below this is a link to 'Learn more'. The main area lists ASGs with their status and expanded rules:

- public_networks**
running, staging
 - 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
- dns**
running, staging
- sshfs-service**
running
- p-mysql**
running

Configuring Spring Boot Actuator Endpoints for Apps Man

Page last updated:

The Apps Manager UI supports several production ready-endpoints from Spring Boot Actuator. This topic describes how you can configure your app to display data from the Actuator endpoints in Apps Manager.

 **Note:** This feature requires Spring Boot v1.5 or later.

For information about how you view the data from these endpoints in Apps Manager, see the [Using Spring Boot Actuator Endpoints in Apps Manager](#) topic.

Overview: Spring Boot Actuators

Spring Boot Actuator provides services for monitoring and managing your Spring Boot app. You can add Actuators to your app project and access them through HTTP endpoints.

Apps Manager supports the following endpoints:

- `/info`: Exposes details about application environment, git, and build
- `/health`: Shows health status or detailed health information over a secure connection
- `/loggers`: Lists and allows modification of the levels of the loggers in an application

For more information about Spring Boot Actuators, see the [Spring Boot Actuator documentation](#).

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.

Follow the instructions 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")
}
```

Configure the Health Actuator

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.

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 for your project build 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 "1.4.6"
}
```

Using Spring Boot Actuators with Apps Manager

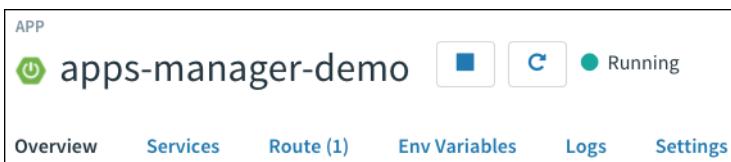
This document describes how to view and manage app information from Spring Boot Actuator in Apps Manager.

Prerequisites

The Apps Manager integration with Spring Boot Actuator requires the following:

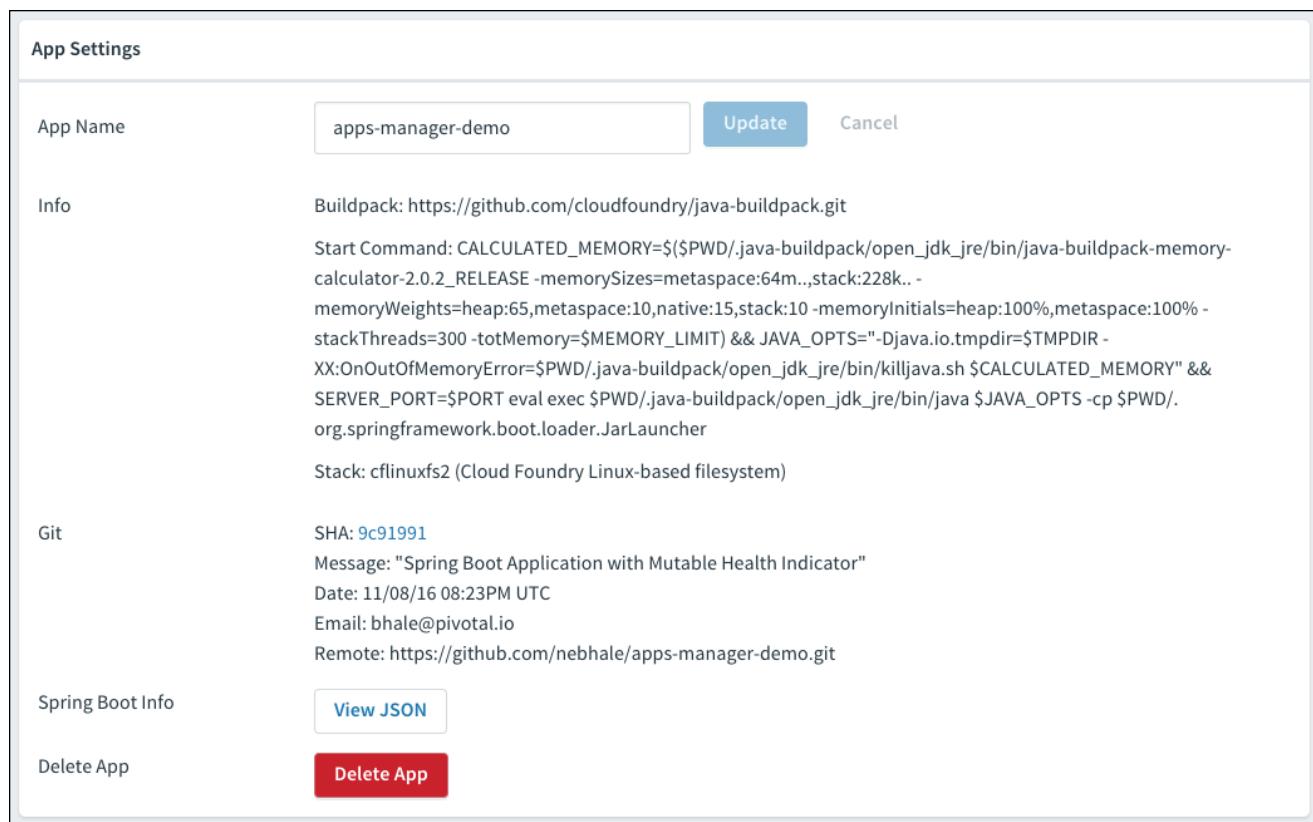
- Spring Boot v1.5 or later
- Completing the procedures in [Configure Spring Boot Actuator Endpoints for Apps Manager](#).

After you configure your app, Apps Manager displays the Spring Boot logo next to its name 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:



App Settings	
App Name	<input type="text" value="apps-manager-demo"/> <button>Update</button> <button>Cancel</button>
Info	Buildpack: https://github.com/cloudfoundry/java-buildpack.git Start Command: CALCULATED_MEMORY=\$(\$PWD/.java-buildpack/open_jdk_jre/bin/java-buildpack-memory-calculator-2.0.2_RELEASE -memorySizes=metaspace:64m.,stack:228k.. -memoryWeights=heap:65,metaspace:10,native:15,stack:10 -memoryInitials=heap:100%,metaspace:100% -stackThreads=300 -totMemory=\$MEMORY_LIMIT) && JAVA_OPTS="-Djava.io.tmpdir=\$TMPDIR -XX:OnOutOfMemoryError=\$PWD/.java-buildpack/open_jdk_jre/bin/killjava.sh \$CALCULATED_MEMORY" && SERVER_PORT=\$PORT eval exec \$PWD/.java-buildpack/open_jdk_jre/bin/java \$JAVA_OPTS -cp \$PWD/.org.springframework.boot.loader.JarLauncher Stack: cflinuxfs2 (Cloud Foundry Linux-based filesystem)
Git	SHA: 9c91991 Message: "Spring Boot Application with Mutable Health Indicator" Date: 11/08/16 08:23PM UTC Email: bhale@pivotal.io Remote: https://github.com/nebhale/apps-manager-demo.git
Spring Boot Info	View JSON
Delete App	Delete App

In the upper right of the app page, Apps Manager also displays the SHA of your app code repository from the latest build:



Git: 9c91991
Buildpack: https://github.com/cl...
Routes: 1

View App Health

To view the health-check data that your app sends to its `/health` Actuator endpoints, see the **Instances** section:

The screenshot shows the 'Instances' section of the Pivotal Apps Manager. It displays a table with one row for an application instance. The columns are: # (with a dropdown arrow), APP HEALTH (Up), CPU (0%), MEMORY (324.91 MB), DISK (136.91 MB), and UPTIME (7 hr 59 min). Below the table, there's a 'Health Check' section with a 'View JSON' link. It shows two metrics: 'mutable' with status UP and 'diskSpace' with status UP, total 1056858112, free: 913297408, and threshold: 10485760.

Manage Log Levels in Apps Manager

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 the **Configure Logging Levels** button.

The screenshot shows the 'Logs' tab of the Pivotal Apps Manager. At the top right, it shows Git: c2d4c3e and Buildpack: https://github.com/cl... Below that is a 'Configure Logging Levels' button with a play icon. The main area shows a log entry: response_time:0.006/33497 app_id:Tb4/8dT4-3cba-420b-a15b-220c430e443T app_index:1 server-demo.al-app.cf-app.com - [06/12/2016:22:15:18.854 +0000] "GET

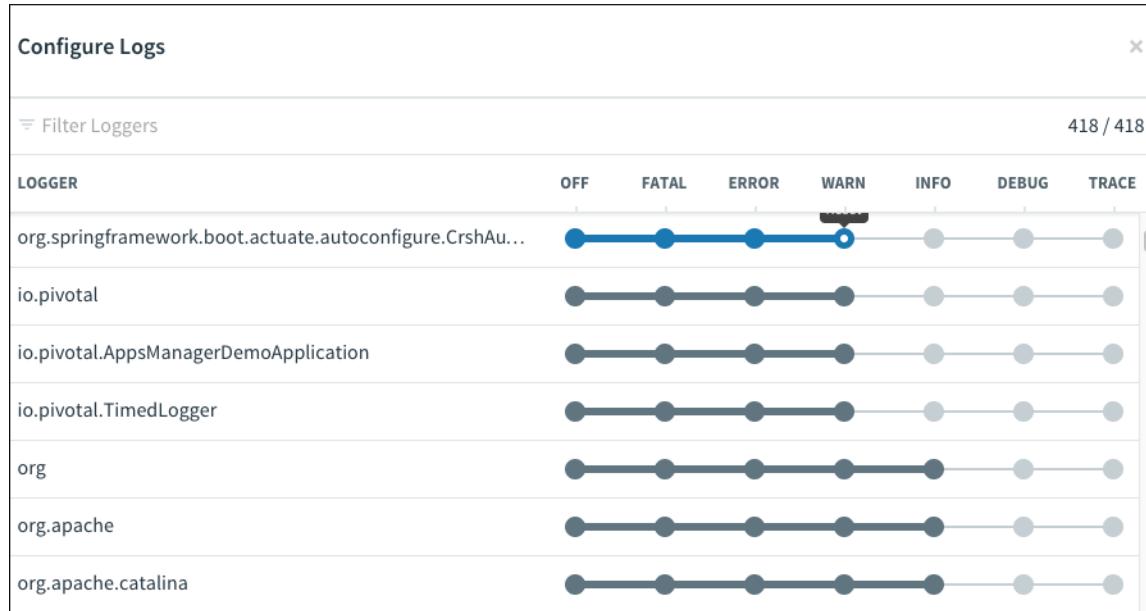
Apps Manager displays the default log level for each logger in gray.

The screenshot shows the 'Configure Logs' dialog. At the top left is a 'Configure Logs' button and a close 'x' button. At the top right, it says 418 / 418. Below that is a 'Filter Loggers' input field. The main area is a table where loggers are listed on the left and log levels are set on the right. The log levels are represented by sliders: OFF, FATAL, ERROR, WARN, INFO, DEBUG, and TRACE. Most loggers have their log level set to INFO or higher. The loggers listed are: io, io.pivot, io.pivot.AppsManagerDemoApplication, io.pivot.TimedLogger, org, org.apache, org.apache.catalina, org.apache.catalina.core, and org.apache.catalina.core.ContainerBase. At the bottom right is a 'Close' button.

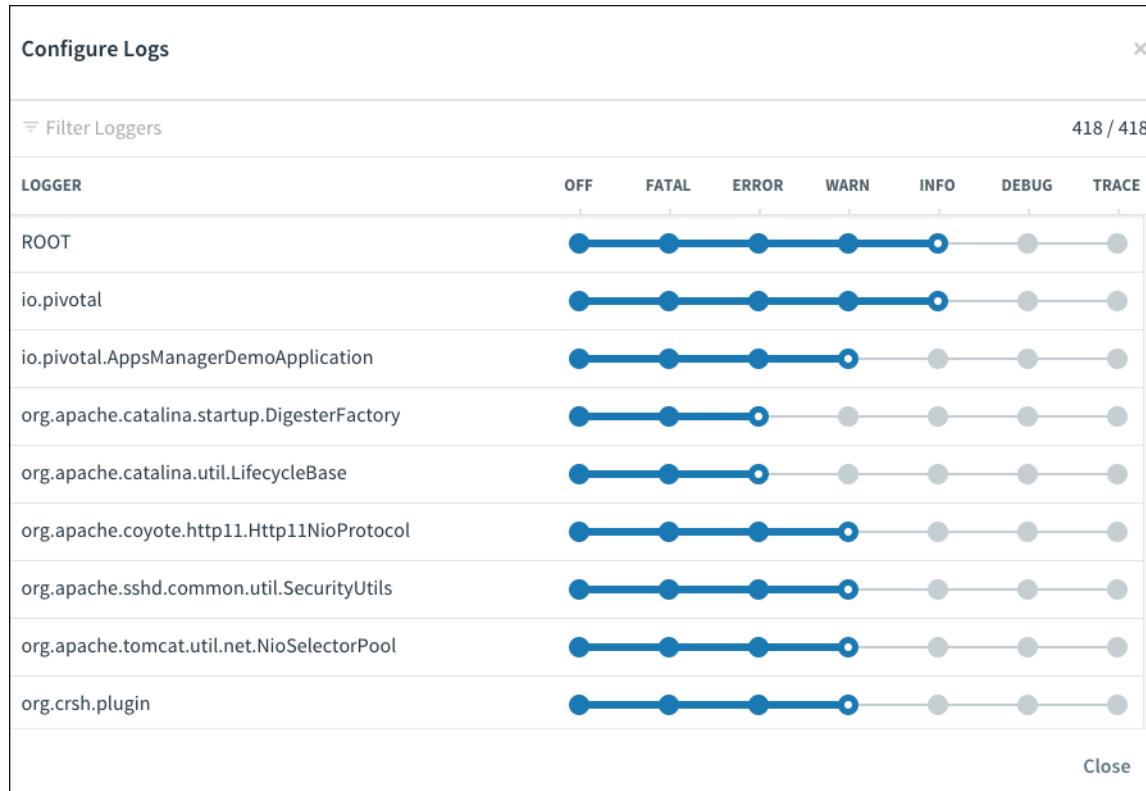
You can modify the log level for any logger by clicking the desired level in the logger row as in the image below. Whenever you set a log level, the following happens:

- The log level becomes blue to indicate that it is user-configured.
- Each child namespace of the logger inherits the log level.

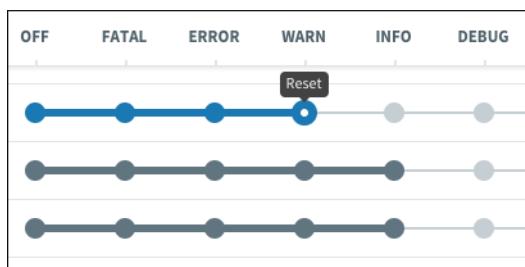
 **Note:** You can manually set any of the child loggers to override this inheritance.



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 what loggers you see using the **Filter Loggers** textbox.

Configure Logs	
= org	
LOGGER	
org	
org.apache	
org.apache.catalina	
org.apache.catalina.core	

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](#)
- [Getting Started with the cf CLI](#)
- [Using the cf CLI with an HTTP Proxy Server](#) ↗
- [Using the cf CLI with a Self-Signed Certificate](#)
- [Using cf CLI Plugins](#)
- [Developing cf CLI Plugins](#)
- [Cloud Foundry CLI Reference Guide](#)

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 Elastic Runtime. 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 Elastic Runtime 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 Elastic Runtime 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 Elastic Runtime 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 an HTTP Proxy Server

Page last updated:

If you have an HTTP 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`

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:

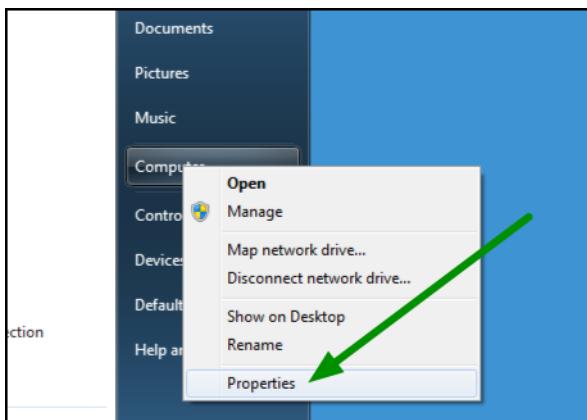
```
$ export https_proxy=http://my.proxyserver.com:8080
```

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:

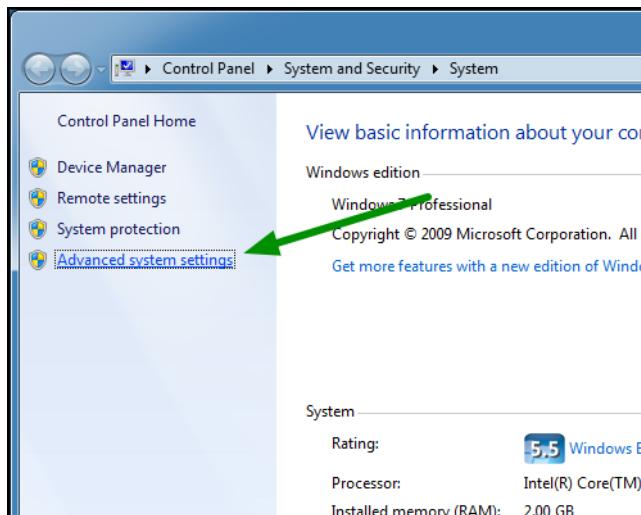
```
https_proxy=http://username:password@hostname:port  
export $https_proxy
```

Setting https_proxy in Windows

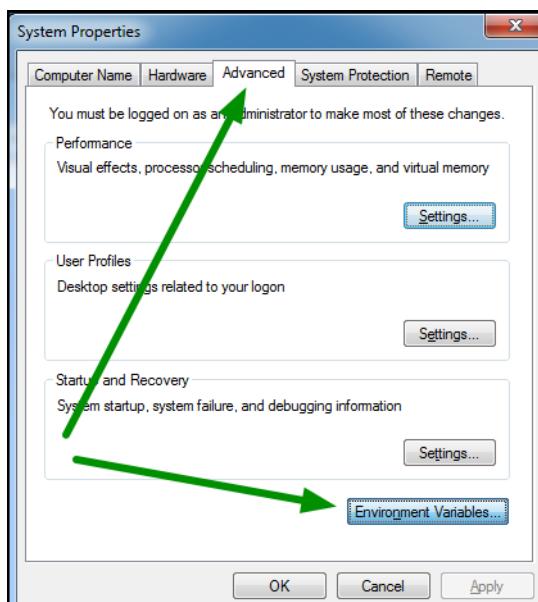
1. Open the Start menu. Right-click **Computer** and select **Properties**.



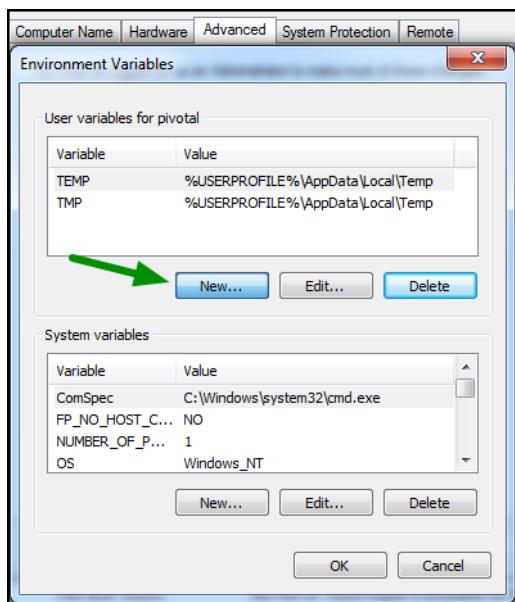
2. In the left pane of the System window, click **Advanced system settings**.



- In the System Properties window, select the **Advanced** tab, then click **Environment Variables**.



- 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 Cloud Foundry (CF) 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 CF operator or from the deployment manifest. Review the [Securing Traffic into Cloud Foundry](#) topic for more information about how to retrieve certificates from the deployment manifest.

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
```

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.

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 plugin names and commands, but not case-sensitive binary filenames.

Prerequisites

Using plugins requires cf CLI v.6.7 or higher. Refer to the [Installing the cf CLI](#) topic for information about downloading, installing, and uninstalling the cf CLI.

Changing the Plugin Directory

By default, the cf CLI stores plugins in `$HOME/.cf/plugins` on your workstation. To change the root directory of this path from `$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 must first 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 PLUGIN` and `PLUGIN COMMANDS` 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.

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 Repo

Run `cf add-plugin-repo REPO_NAME URL` to add a plugin repo.

Example:

```
$ cf add-plugin-repo CF-Community https://plugins.cloudfoundry.org
OK
https://plugins.cloudfoundry.org/list added as 'CF-Community'
```

Listing Available Plugin Repos

Run `cf list-plugin-repos` to view your available plugin repos.

Example:

```
$ cf list-plugin-repos
OK
Repo Name      Url
CF-Community   https://plugins.cloudfoundry.org
```

Listing All Plugins by Repo

Run `cf repo-plugins` to show all plugins from all available repos.

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 repo](#).

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 repo](#). 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 CLI Commands

Invoke 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 CLI command fails.

For more information, see the [calling CLI commands example](#).

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

6.25.0+787326d.2017-02-28

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 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 may cause 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 the running application Instance at the given index and instantiate a new instance of the application with the same index
run-task	Run a one-off task on an app
tasks	List tasks of an app

Command	Description
terminate-task [x]	Terminate a running task of an app
events [x]	Show recent app events
files [x]	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 [x]	Tail or show recent logs for an app
env [x]	Show all env variables for an app
set-env [x]	Set an env variable for an app
unset-env [x]	Remove an env variable
stacks [x]	List all stacks (a stack is a pre-built file system, including an operating system, that can run apps)
stack [x]	Show information for a stack (a stack is a pre-built file system, including an operating system, that can run apps)
copy-source [x]	Copies the source code of an application to another existing application (and restarts that application)
create-app-manifest [x]	Create an app manifest for an app that has been pushed successfully
get-health-check [x]	Show the type of health check performed on an app
set-health-check [x]	Change type of health check performed on an app
enable-ssh [x]	Enable ssh for the application
disable-ssh [x]	Disable ssh for the application
ssh-enabled [x]	Reports whether SSH is enabled on an application container instance
ssh [x]	SSH to an application container instance

Services

Command	Description
marketplace [x]	List available offerings in the marketplace
services [x]	List all service instances in the target space
service [x]	Show service instance info
create-service [x]	Create a service instance
update-service [x]	Update a service instance
delete-service [x]	Delete a service instance
rename-service [x]	Rename a service instance
create-service-key [x]	Create key for a service instance
service-keys [x]	List keys for a service instance
service-key [x]	Show service key info
delete-service-key [x]	Delete a service key
bind-service [x]	Bind a service instance to an app
unbind-service [x]	Unbind a service instance from an app
bind-route-service [x]	Bind a service instance to an HTTP route
unbind-route-service [x]	Unbind a service instance from an HTTP route
create-user-provided-service [x]	Make a user-provided service instance available to CF apps
update-user-provided-service [x]	Update user-provided service instance

Orgs

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)

Buildpacks

Command	Description

Buildpacks	Description
create-buildpack	List all buildpacks 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
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

Command	Description
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

Feature Flags

Command	Description
feature-flags	Retrieve list of feature flags with status of each flag-able feature
feature-flag	Retrieve an individual feature flag with status
enable-feature-flag	Enable the use of a feature so that users have access to and can use the feature
disable-feature-flag	Disable the use of a feature so that users have access to and can use the 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 all available plugin commands
install-plugin	Install CLI plugin
uninstall-plugin	Uninstall the plugin defined in command argument

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

Variable	Description
HTTP_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

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](#)
-

Deploying and Managing Applications

- [Deploy an Application](#)
 - [Deploy a Large Application](#)
 - [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](#)
-

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](#)
- [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.

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`.

Consider whether 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 or configuration files with the following file extensions:

- `.cfignore`
- `_darcs`
- `.DS_Store`
- `.git`
- `.gitignore`
- `.hg`
- `/manifest.yml`
- `.svn`

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, the best practice is to exclude them using a `.cfignore` file. (`.cfignore` is similar to git's `.gitignore`, which allows you to exclude files and directories from git tracking.) Especially with a large application, uploading unnecessary files slows down application deployment.

Specify the files or file types you wish to exclude from upload in a text file, named `.cfignore`, in the root of your application directory structure. For example, these lines exclude the "tmp" and "log" directories.

```
tmp/  
log/
```

The file types you will want to exclude vary, based on the application frameworks you use. The `.gitignore` templates for common frameworks, available at <https://github.com/github/gitignore>, are a useful starting point.

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 complete 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 Elastic Runtime 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 Elastic Runtime. 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, without spaces, 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 with file extensions `.svn`, `.git`, and `.dars`. To exclude other files from upload, specify them in a `.cfignore` file in the directory where you run the push command. This technique is similar to using a `.gitignore` file. 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 Elastic Runtime, 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 > Elastic Runtime > 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:** Elastic Runtime 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 the cf CLI waits for an app to start. When you use this flag, the cf CLI ignores any app start timeout value set in the manifest or in the `CF_STARTUP_TIMEOUT` environment variable. 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. Server-side timeouts must be changed 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 Elastic Runtime.

Setting	Note
App Package Size	Maximum: Set in Ops Manager > Elastic Runtime > Application Developer Controls
Authorization Token Grace Period	Default: 20 minutes, minimum
<code>CF_STAGING_TIMEOUT</code>	cf CLI environment variable Default: 15 minutes
<code>CF_STARTUP_TIMEOUT</code>	cf CLI environment variable Default: 5 minutes
	App start timeout maximum

<code>cf push -t TIMEOUT</code>	Default: 60 seconds
Disk Space Allocation	Default: 1024 MB
Internet Connection Speed	Recommended Minimum: 874 KB/s

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 need to 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 Elastic Runtime, 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 Elastic Runtime, see the [HTTP Routing](#) topic.

Routes

The Elastic Runtime 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
<code>http://myapp.shared-domain.example.com</code>	<code>myapp.shared-domain.example.com</code>
<code>http://myapp.shared-domain.example.com/contact</code>	<code>myapp.shared-domain.example.com</code>
<code>http://myapp.shared-domain.example.com/products</code>	<code>myapp.shared-domain.example.com/products</code>
<code>http://myapp.shared-domain.example.com/products/123</code>	<code>myapp.shared-domain.example.com/products</code>
<code>http://products.shared-domain.example.com</code>	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, Elastic Runtime 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, Elastic Runtime 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 Elastic Runtime 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 Elastic Runtime, 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 Elastic Runtime 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 Elastic Runtime 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 Elastic Runtime 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`, Elastic Runtime 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:

- Elastic Runtime routes requests to `http://store.shared-domain.example.com/products` to the `products` app.
- Elastic Runtime routes requests to `http://store.shared-domain.example.com/orders` to the `orders` app.
- Elastic Runtime routes requests to `http://store.shared-domain.example.com` to the `storefront` app.

Elastic Runtime 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: Elastic Runtime 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 Elastic Runtime 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, Elastic Runtime 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

Elastic Runtime 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

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 `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-a86b92f2df0c` and mapped to route `myapp.private-domain.example.com`.

```
$ curl myapp.private-domain.example.com -H "X-Cf-App-Instance: 5cdc7595-2e9b-4f62-8d5a-a86b92f2df0c: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 Elastic Runtime. 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, Elastic Runtime only supports routing of HTTP requests to applications.

Shared Domains

Admins manage shared domains, which are available to users in all orgs of a Elastic Runtime 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 Elastic Runtime 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 Elastic Runtime 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 Elastic Runtime by using your DNS provider.

Mapping Domains to Your Custom Domain

To associate a registered domain name with a domain on Elastic Runtime, configure a CNAME record with your DNS provider, pointing at any shared domain offered in Elastic Runtime.

Mapping a Single Domain to Your Custom Domain

To map a single domain to a custom domain to Elastic Runtime, 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 Elastic Runtime
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 Elastic Runtime
*.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 Elastic Runtime 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 Elastic Runtime.

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 Elastic Runtime.

Record	Name	Target	Note
URL or Forward	private-domain.example.com	www.private-domain.example.com	This method results in a <code>301 permanent redirect</code> 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 Elastic Runtime. 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:

Application manifests tell `cf push` what to do with applications. This includes everything from how many instances to create and how much memory to allocate to what services applications should use.

A manifest can help you automate deployment, especially of multiple applications at once.

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.

```
$ cf push  
Using manifest file /path_to_working_directory/manifest.yml
```

If your manifest is located elsewhere, use the `-f` option to provide the path to the filename.

```
$ cf push -f ./some_directory/some_other_directory/alternate_manifest.yml  
Using manifest file /path_to_working_directory/some_directory/some_other_directory/alternate_manifest.yml
```

If you provide a path with no filename, the filename must be `manifest.yml`.

```
$ cf push -f ./some_directory/some_other_directory/  
Using manifest file /path_to_working_directory/some_directory/some_other_directory/manifest.yml
```

Example Manifest

You can deploy applications without ever using a manifest. The benefits manifests may provide include consistency and reproducibility. When you want applications to be portable between different clouds, manifests may prove especially useful.

Manifests are written in YAML. The manifest below illustrates some YAML conventions, as follows:

- The manifest may begin 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: nifty-gui  
  memory: 512M  
  host: nifty
```

A minimal manifest requires only an application `name`. To create a valid minimal manifest, remove the `memory` and `host` 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 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`.

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`.

hosts

Use the `hosts` attribute to provide multiple hostnames, or subdomains. Each hostname generates a unique route for the app. `hosts` can be used in 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 newer 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 none`.

In the older Droplet Execution Agent (DEA) architecture, `cf set-health-check APP_NAME none` is unnecessary because `no-route` causes the DEAs to skip the port health-check on app startup.

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
<code>--no-route</code>	All declared routes are ignored.
<code>-d</code>	Overrides DOMAIN part of all declared HTTP and TCP routes.
<code>--hostname</code> , <code>-n</code>	Sets or overrides HOSTNAME in all HTTP routes. It has no impact on TCP routes.
<code>--route-path</code>	Sets or overrides the PATH in all HTTP routes. It has no impact on TCP routes.
<code>--random-route</code>	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.

For example:

```
---  
...  
timeout: 80
```

You can increase the timeout length for very large apps that require more time to start. The default timeout is 60 seconds, with an upper bound of 180 seconds.

 **Note:** Administrators can set the upper bound of the `maximum_health_check_timeout` property to any value. Any changes to Cloud Controller properties in the deployment manifest require running `bosh deploy`.

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 them in a single manifest. In doing so, you need to pay extra attention to directory structure and path lines in the 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 called `base-manifest.yml` begins like this:

```
---
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: springstock
  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 <code>port</code> 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
<code>http</code>	The app can provide an <code>HTTP 200</code> response.	The <code>http</code> health check performs a GET request to the configured http endpoint. When the health check receives an <code>HTTP 200</code> response, the app is declared healthy. We recommend using the <code>http</code> 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.
<code>port</code>	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 <code>port</code> health check. The TCP connection must be established within 1 second to be considered healthy.
<code>process</code>	The app does not support TCP connections (for example, a worker).	For a <code>process</code> 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.

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 M` 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 Elastic Runtime.

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)
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 D, [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
```

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": "fa05c1a9-0fc1-4fb1-bae1-139850dec7a3",
    "application_name": "my-app",
    "application_uris": [
      "my-app.192.0.2.34.xip.io"
    ],
    "application_version": "fb8bcc6-8d58-479e-bcc7-3b4ce5a7f0ca",
    "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": "fb8bcc6-8d58-479e-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](#).

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. Available only to instances on Diego Cells.

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=1.2.3.4

CF_INSTANCE_PORT

The external, or *host-side*, port corresponding to the internal, or *container-side*, port with value [PORT](#). For instances on Diego, 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. On DEAs, these internal and external values are the same, but on Diego Cells they may differ.

Example: CF_INSTANCE_PORTS=[{external:61045,internal:5678},{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/exampledbs"  
      }  
    }  
  ]  
}
```

Based on this `VCAP_SERVICES`, CF creates the following `DATABASE_URL` environment variable:

```
DATABASE_URL = postgres://exampleuser:examplepass@babar.elephantsql.com:5432/exampledbs
```

HOME

Root folder for the deployed application.

Example: `HOME=/home/vcap/app`

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=61857`

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_HOST

The IP address of the host. Deprecated: the DEAs set this to be `0.0.0.0`, and Diego Cells do not provide this environment variable.

Example: `VCAP_APP_HOST=0.0.0.0`

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
<code>application_id</code>	GUID identifying the application.
<code>application_name</code>	The name assigned to the application when it was pushed.
<code>application_uris</code>	The URIs assigned to the application.
<code>application_version</code>	GUID identifying a version of the application. Each time an application is pushed or restarted, this value is updated.
<code>host</code>	Deprecated. IP address of the application instance.
<code>instance_id</code>	Unique ID that identifies the application instance. For instances running on Diego, this is identical to the CF_INSTANCE_GUID variable.
<code>instance_index</code>	Index number of the instance. Identical to the CF_INSTANCE_INDEX variable.
<code>limits</code>	The memory, disk, and number of files permitted to the instance. Memory and disk 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.
<code>name</code>	Identical to <code>application_name</code> .
<code>port</code>	Port of the application instance. Identical to the <code>PORT</code> variable.
<code>space_id</code>	GUID identifying the application's space.
<code>start</code>	Human-readable timestamp for the time the instance was started. Not provided on Diego Cells.
<code>started_at</code>	Identical to <code>start</code> . Not provided on Diego Cells.
<code>started_at_timestamp</code>	Unix epoch timestamp for the time the instance was started. Not provided on Diego Cells.
<code>state_timestamp</code>	Identical to <code>started_at_timestamp</code> . Not provided on Diego Cells.
<code>uris</code>	Identical to <code>application_uris</code> . You must ensure that both <code>application_uris</code> and <code>uris</code> are set to the same value.
<code>users</code>	Deprecated. Not provided on Diego Cells.
<code>version</code>	Identical to <code>application_version</code> .

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
<code>name</code>	The name assigned to the service instance by the user.
<code>label</code>	The name of the service offering.
<code>tags</code>	An array of strings an app can use to identify a service instance.
<code>plan</code>	The service plan selected when the service instance was created.
<code>credentials</code>	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
<code>running-environment-variable-group</code> or <code>revg</code>	Retrieves the contents of the running environment variable group.
<code>staging-environment-variable-group</code> or <code>sevg</code>	Retrieves the contents of the staging environment variable group.
<code>set-staging-environment-variable-group</code> or <code>ssevg</code>	Passes parameters as JSON to create a staging environment variable group.
<code>set-running-environment-variable-group</code> or <code>srevg</code>	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.

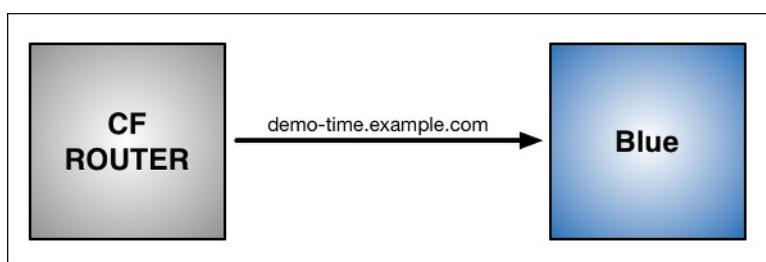
Step 1: Push an App

Use the 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.



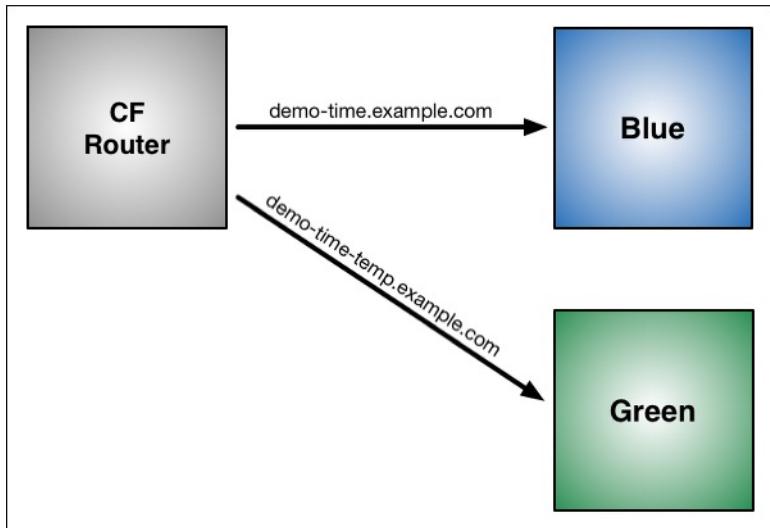
Step 2: Update App and Push

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.



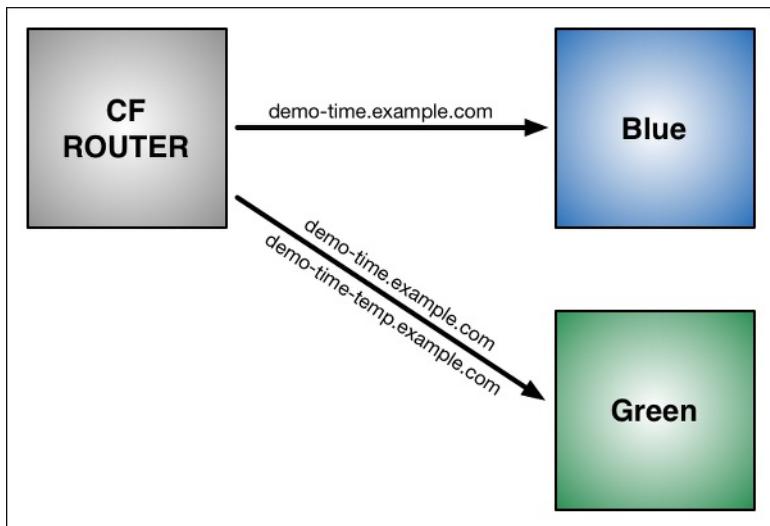
Step 3: Map Original Route to Green

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.



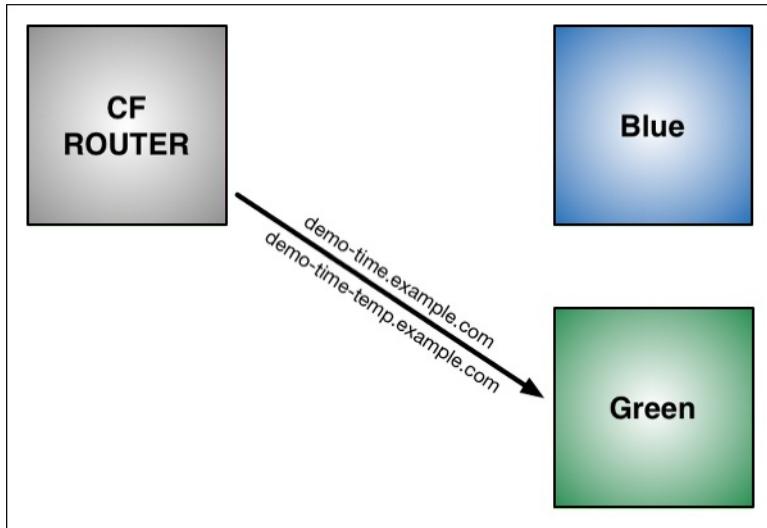
Step 4: Unmap Route to Blue

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
```

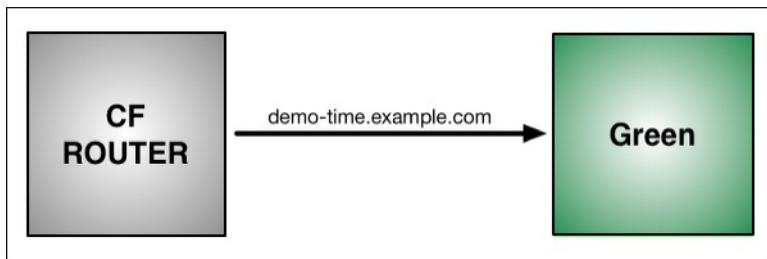
After `cf unmap-route` command:

- 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 help` 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 Elastic Runtime. 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 the [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

Elastic Runtime requires that each app that you deploy has a unique URL. Otherwise, the new app URL collides with an existing app URL and Elastic Runtime 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":>0, "index":>0, "reason":>"CRASHED",
"exit_description":>"2 error(s) occurred:\n* 1 error(s)
occurred:\n* Exited with status 6\n* 2 error(s)
occurred:\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

Elastic Runtime 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 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 of 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. See <https://github.com/cloudfoundry/errors> for a list of Cloud Foundry errors. Shows that an app instance exited; for more detail, look in the application logs.
- `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 Elastic Runtime 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 in order to troubleshoot.

SSH Access Control Hierarchy

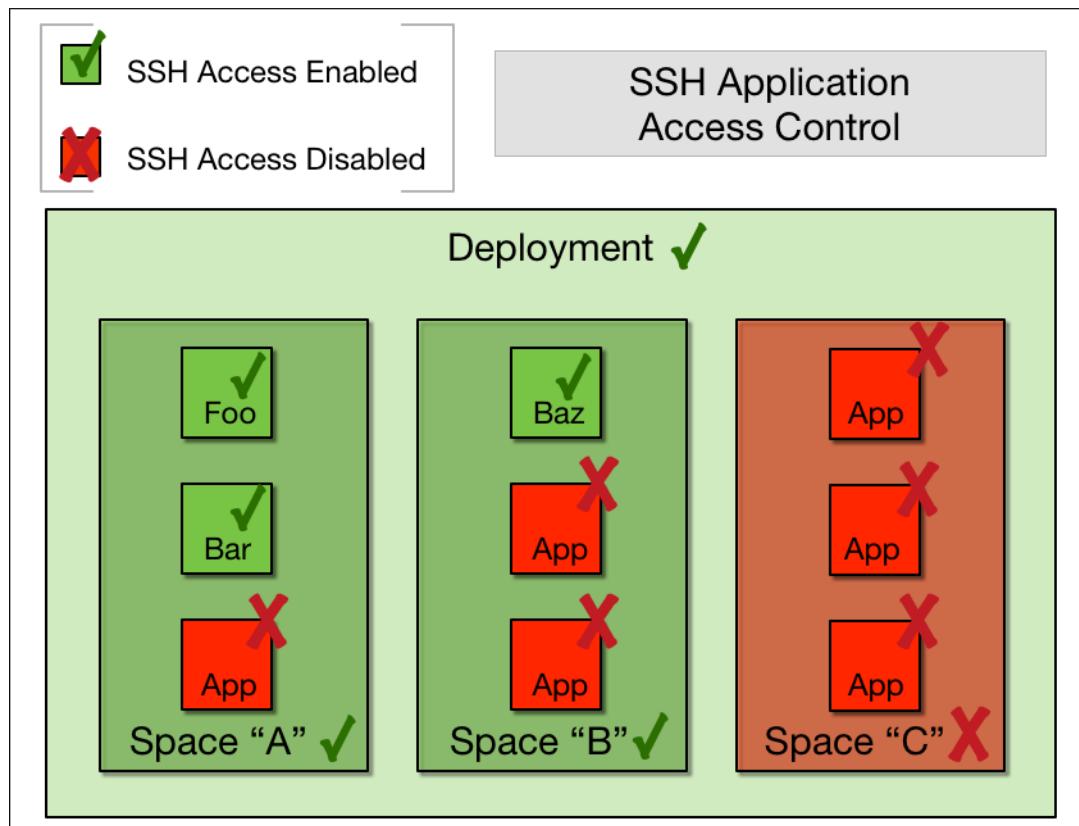
Operators, space managers, and space developers can configure SSH access for Elastic Runtime, spaces, and apps as described in this table:

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 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 Elastic Runtime

Pivotal Cloud Foundry deployments control SSH access to apps at the Elastic Runtime 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 Elastic Runtime 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
<code>cf enable-ssh</code> <code>cf disable-ssh</code> <code>cf allow-space-ssh</code> <code>cf disallow-space-ssh</code>	Enable and Disable SSH Access
<code>cf ssh-enabled</code> <code>cf space-ssh-allowed</code>	Check SSH Access Permissions
<code>cf ssh</code>	Securely log into an application container
<code>cf ssh-code</code>	Enable secure log in to an application container using non-CF SSH tools like <code>ssh</code>, <code>scp</code>, and <code>sftp</code>

Enabling and Disabling SSH Access

A cloud operator can deploy Elastic Runtime 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.

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
```

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`.

- 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
```

- 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",
...
}
```

- 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
```

- 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
```

- 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.

C_{ONTAINER}_P_{ORT} (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.

H_{OST}_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 -f .  
1
```

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.

P_{RIVATE}_K_{EY} (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 `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 Elastic Runtime applications which reflect the policies that are in effect.

Accessing Services with SSH

Page last updated:

This page assumes you are using [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 utilize its SSH and port forwarding features to communicate with the service instance through the app container. The technique outlined below works with any TCP service, such as MySQL or Redis.

Create a Service Instance

In your terminal window, log in to your deployment with [cf login](#).

1. 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
```

2. 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 Elastic Runtime.

 **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:** In order to enable SSH access to your app, SSH access must also be enabled for both the space that contains the app and Elastic Runtime. See the [Application SSH Overview](#) topic for further details.

Create Your Service Key

To establish SSH access to your service instance, you need to 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&password=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` indicates to `mysql` to connect to your local machine.
- `-p` indicates to `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` indicates to `mysql` to connect on port 63306.

Trusted System Certificates

Page last updated:

The Cloud Foundry Administrator can deploy a set of trusted system certificates to be made 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 has configured these certificates, they will be 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 will also be installed directly in the `/etc/ssl/certs` directory, and so will be available automatically to libraries such as `openssl` that respect that trust store.

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](#)

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 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
```

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.

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` 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 of 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 manually with service keys. You can use service keys to connect to a service instance from a local client, or from an app that is not deployed by Cloud Foundry. Create, list, retrieve, and delete service keys using the Cloud Foundry Command Line Interface (cf CLI).

 **Note:** Not all services support service keys. Some services support credentials through [application binding](#) only.

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 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...
```

```
e3696fc...-7a8f-437f-8692-436558e45c7b
```

```
OK
```

Delete 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:

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 cf CLI will prompt you for each parameter value.

```
cf cups SERVICE_INSTANCE -p "host, port, dbname, username, password"
```

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.

In order to proxy requests to the user-provided route service, you will need to 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:** Elastic Runtime 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 `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 stack you want to use.
2. Click Logstash **Configuration**.
3. Note your Logstash **Endpoint**.
4. Note your TCP or UDP **Port** (not the syslog port).
5. Create the log drain service in Cloud Foundry.

```
$ cf cups logit-drain -l syslog://ENDPOINT:PORT
```

6. Bind the service to an app.

```
$ 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**.

The screenshot shows the Papertrail Dashboard. At the top, there are buttons for "Add Systems" and "Create Group". Below that, a message box says: "Let's aggregate some logs. Add your first system in about 45 seconds, or take a tour." There is a small icon of a person with a gear next to the message.

2. Click the **Other** link.

The screenshot shows the "Setup Systems" page. It has a header "Setup Systems" and a sub-header "Your systems will log to **logs2.papertrailapp.com:14608**". At the bottom, there is a link "» Other situations: Port 514 | Other".

3. Select **I use Cloud Foundry**, enter a name, and click **Save**.

Choose your situation:

- A My syslog only uses the default port**

GNU syslogd and some embedded devices will only log to port 514. A few old Linux distro versions use GNU syslogd (mostly CentOS and Gentoo).

- B I use Cloud Foundry**

Register each app separately. Use Heroku? [Here's how.](#)

- C My system's hostname changes**

In rare cases, one system may change hostnames frequently. For example, a roaming laptop which sets its hostname based on DHCP (and roams across networks).

We'll provide an app-specific syslog drain and step-by-step setup for [Cloud Foundry](#).

Let's create a destination for this app.

What should we call it?

CloudFoundry

Alphanumeric. Does not need to match app name.

Save →

- Record the URL with port that is displayed after creating the system.

CloudFoundry will log to `logs.papertrailapp.com:36129`.

- Create the log drain service in Cloud Foundry.

```
$ cf cups my-logs -l syslog-tls://logs.papertrailapp.com:PORT
```

- Bind the service to an app.

```
$ cf bind-service APPLICATION-NAME my-logs
```

- Restart the app.

```
$ cf restart APPLICATION-NAME
```

After a short delay, logs begin to flow automatically.

- Once Papertrail starts receiving log entries, the view automatically updates to the logs viewing page.

```
All Systems - Dashboard Events Account Help - Me -
Skipping auto-reconfiguration.
Mar 05 14:57:36 CloudFoundry web@993-4088-8718-186f147f5e73[App/1] Mar 05, 2014 10:57:36 PM
org.cloudfoundry.reconfiguration.AbstractServiceConfigurer configure
Mar 05 14:57:36 CloudFoundry web@993-4088-8718-186f147f5e73[App/1] INFO: No beans of type org.springframework.data.mongodb.MongoDbFactory found in application context. Skipping auto-reconfiguration.
Mar 05 14:57:36 CloudFoundry web@993-4088-8718-186f147f5e73[App/1] Mar 05, 2014 10:57:36 PM
org.cloudfoundry.reconfiguration.AbstractServiceConfigurer configure
Mar 05 14:57:36 CloudFoundry web@993-4088-8718-186f147f5e73[App/1] INFO: No beans of type org.springframework.data.redis.connection.RedisConnectionFactory found in application context. Skipping auto-reconfiguration.
Mar 05 14:57:36 CloudFoundry web@993-4088-8718-186f147f5e73[App/1] Mar 05, 2014 10:57:36 PM
org.cloudfoundry.reconfiguration.AbstractServiceConfigurer configure
Mar 05 14:57:36 CloudFoundry web@993-4088-8718-186f147f5e73[App/1] INFO: Class org.springframework.orm.rabbit.connection.ConnectionFactory not in classpath. Skipping auto-reconfiguration for it
Mar 05 14:57:36 CloudFoundry web@993-4088-8718-186f147f5e73[App/1] 22:57:36,688 INFO RequestMappingHandlerMapping:181 - Mapped "/{albums}/{id}" [methods={DELETE}],params=[],headers=[],consumes[],produces[],custom={}
org.cloudfoundry.samples.music.web.controllers.AlbumController.deleteById(id:long,String)
Mar 05 14:57:36 CloudFoundry web@993-4088-8718-186f147f5e73[App/1] 22:57:36,688 INFO RequestMappingHandlerMapping:181 - Mapped "/{albums}/{id}/{path}" [methods={PUT}],params=[],headers[],consumes[],produces[],custom={}
org.cloudfoundry.samples.music.web.controllers.AlbumController.update(org.cloudfoundry.samples.music.domain.Album)
Mar 05 14:57:36 CloudFoundry web@993-4088-8718-186f147f5e73[App/1] 22:57:36,688 INFO RequestMappingHandlerMapping:181 - Mapped "/{albums}/{id}/{path}/{path2}" [methods={PUT}],params=[],headers[],consumes[],produces[],custom={}
org.cloudfoundry.samples.music.web.controllers.AlbumController.update(org.cloudfoundry.samples.music.domain.Album)
Mar 05 14:57:36 CloudFoundry web@993-4088-8718-186f147f5e73[App/1] 22:57:36,688 INFO RequestMappingHandlerMapping:181 - Mapped "({@Info},methods[],params[],headers[],consumes[],produces[],custom{})" onto public org.cloudfoundry.samples.music.domain.Album
Mar 05 14:57:36 CloudFoundry web@993-4088-8718-186f147f5e73[App/1] 22:57:36,688 INFO RequestMappingHandlerMapping:181 - Mapped "({@Service},methods[],params[],headers[],consumes[],produces[],custom{})" onto public org.cloudfoundry.cloud.ServiceInfoController.showServiceInfo()
Mar 05 14:57:36 CloudFoundry web@993-4088-8718-186f147f5e73[App/1] 22:57:36,688 INFO RequestMappingHandlerMapping:181 - Mapped "[{}/env],methods[],params[],headers[],consumes[],produces[],custom{}" onto public java.util.List<java.util.Map> java.util.List<java.util.Map>
org.cloudfoundry.samples.music.web.controllers.InfoController.showEnvironment()
Mar 05 14:57:36 CloudFoundry web@993-4088-8718-186f147f5e73[App/1] 22:57:36,726 INFO SimpleHandlerMapping:315 - Root mapping to handler of type [class org.springframework.web.servlet.mvc.ParameterizableViewController]
Mar 05 14:57:36 CloudFoundry web@993-4088-8718-186f147f5e73[App/1] 22:57:36,726 INFO SimpleHandlerMapping:315 - Mapped URL path [/assets/**] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
Mar 05 14:57:36 CloudFoundry web@993-4088-8718-186f147f5e73[App/1] 22:57:36,731 INFO SimpleHandlerMapping:315 - Mapped URL path [/**] onto handler of type [class org.springframework.web.servlet.resource.DefaultServletHttpRequestHandler]
Mar 05 14:57:37 CloudFoundry web@993-4088-8718-186f147f5e73[App/1] 22:57:37,940 INFO DispatcherServlet:408 - FrameworkServlet 'appServlet': init() completed in 989 ms
Mar 05 14:57:37 CloudFoundry web@993-4088-8718-186f147f5e73[App/1] Mar 05, 2014 10:57:37 PM org.apache.coyote.AbstractProtocol start
Mar 05 14:57:37 CloudFoundry web@993-4088-8718-186f147f5e73[App/1] INFO: Starting ProtocolHandler ["http-bio-62939"]
Mar 05 14:57:37 CloudFoundry web@993-4088-8718-186f147f5e73[App/1] Mar 05, 2014 10:57:37 PM org.apache.catalina.startup.Catalina start
Mar 05 14:57:37 CloudFoundry web@993-4088-8718-186f147f5e73[App/1] INFO: Server startup in 11278 ms
Mar 05 14:59:37 CloudFoundry web@993-4088-8718-186f147f5e73[App/1] 22:59:37,000 INFO org.apache.coyote.AbstractProtocol start
Mar 05 14:59:37 CloudFoundry web@993-4088-8718-186f147f5e73[App/1] Updated app guid 408c960-c093-4088-8718-186f147f5e73 ("instances=>1")
Mar 05 14:59:37 CloudFoundry web@993-4088-8718-186f147f5e73[App/1] Stopping app instance (Index 1) with guid 408c960-c093-4088-8718-186f147f5e73
Mar 05 14:59:37 CloudFoundry web@993-4088-8718-186f147f5e73[App/1] Stopped app instance (Index 1) with guid 408c960-c093-4088-8718-186f147f5e73
```

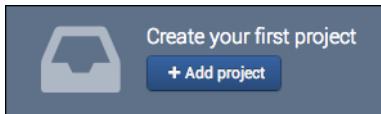
Example: "access denied" (1.2.3.4 OR redis) -sshd
https://papertrailapp.com/groups/5513803/events/centered_on_id=378003402845892611&q=program%3A4aadc960-c093-4088-8718-186f147f5e73%27%5BApp%2F1%5D

See [Streaming Application Logs to Splunk](#) for details.

Splunk Storm

From your Splunk Storm account:

1. Click **Add project**.



2. Enter the project details.

Add project

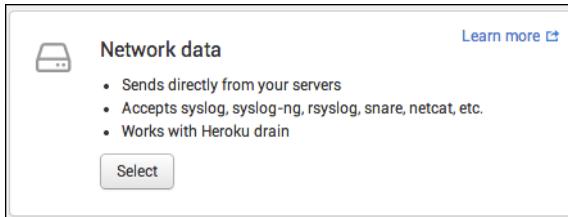
* Project name: Cloud Foundry

* Project time zone: (UTC-0600) America/Chicago

The Storm interface will use this time zone. If data you send to this project does not have a time zone already, it will be assigned this timezone by default. [Learn More](#)

[Cancel](#) [Continue](#)

3. Create a new **input** for **Network data**.



4. Manually enter the external IP addresses your Cloud Foundry administrator assigns to outbound traffic.

Add network data

Storm can receive data from any network port. For example, Storm can accept remote data from [syslog](#), [rsyslog](#), [syslog-ng](#), [snare](#), [netcat](#), etc. or any other application that transmits via TCP or UDP. [Learn more](#)

Authorize your IP address

Automatically
 Manually

5. Note the host and port provided for TCP input.

Authorized network inputs

Storm can receive data from any network port. For example, Storm can accept remote data from [syslog](#), [rsyslog](#), [syslog-ng](#), [snare](#), [netcat](#) or any other application that transmits via TCP or UDP.

1. Authorize your IP address

Authorize automatically
Authorize all IP addresses sending data to this project within the next 15 min.

Authorize manually
Specify the IP address

2. Send data to these ports for this project only

TCP: tcp.k22g-wtfr.data.splunkstorm.com:15486
UDP: udp.k22g-wtfr.data.splunkstorm.com:15486

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**.

The screenshot shows a search interface with the following elements:

- What to Search:** A dropdown menu with options: "266 Events INDEXED", "a minute ago EARLIEST EVENT", and "a few seconds ago LATEST EVENT".
- Data Summary:** A button at the bottom left of the search area.

10. Click the **loggregator** link to view all incoming log entries from Cloud Foundry.

The screenshot shows the "Data Summary" interface with the following details:

- Hosts (1)**: Shows one host named "loggregator".
- Sources (1)**: Shows one source type named "loggregator".
- Sourcetypes (1)**: Shows one sourcetype named "loggregator".
- Filter**: A search bar at the top.
- Table Headers**: Host, Count, Last Update.
- Table Data**: One row for "loggregator" with a count of 26 and last updated on 3/7/14 3:26:01.000 PM.

SumoLogic

Note: SumoLogic uses HTTPS for communication. HTTPS is supported in Cloud Foundry v158 and above.

From your SumoLogic account:

1. Click the **Add Collector** link.

The screenshot shows the "Manage Collectors and Sources" page with the following elements:

- Manage Collectors and Sources**: The main title.
- Links**: Upgrade Collectors, Add Collector, Access Keys.

2. Choose **Hosted Collector** and fill in the details.

The screenshot shows the "Add Collector" dialog with the following details:

- Select a type of collector:**
 - Installed Collector**: Description: "Select to install a Collector in your deployment."
 - Hosted Collector**: Description: "Select to set up a Collector in the Sumo Logic Cloud." This option is highlighted with a yellow background.
- FAQs** section:
 - ▶ What's the difference between an Installed and Hosted Collector?
 - ▶ Where should I install an Installed Collector?
 - ▶ How do I know if I need more than one Installed Collector?
 - ▶ Where does my data go?

Add Collector

Name *	Cloud Foundry
Description	[Empty Text Area]
Category	[Empty Text Area]
Unless overwritten by Source metadata, the Collector will set the Source category of all messages to this value.	
<input type="button" value="Save"/> Cancel	

3. In the new collector's row of the collectors view, click the **Add Source** link.

Manage Collectors and Sources					
Upgrade Collectors Add Collector Access Keys					
Show: All Collectors Running Collectors Stopped Collectors Expand: All None					
Name	Type	Status	Source Category	Sources	Last Hour
Cloud Foundry	Hosted	✓		1	None
					Add Source Edit Delete

4. Select **HTTP** source and fill in the details. Note that you'll be provided an HTTPS url

Select a type of Source:

 Amazon S3	 HTTP
Collects logs from an Amazon S3 bucket.	HTTP receiver that collects logs sent to a specific address.
Name* <input type="text" value="CloudFoundry"/> Maximum name length is 128 characters	
Description <input type="text"/>	
Source Host <input type="text"/>	
Source Category <input type="text"/>	
Advanced Filters	
<input type="button" value="Save"/> Cancel	

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.

Manage Collectors and Sources					
Upgrade Collectors Add Collector Access Keys					
Show: All Collectors Running Collectors Stopped Collectors Expand: All None					
Name	Type	Status	Source Category	Sources	Last Hour
Cloud Foundry	Hosted	✓		1	None
CloudFoundry	HTTP	✓			Show URL Edit Delete

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.

- In the SumoLogic dashboard, click **Manage**, then click **Status** to see a view of log messages received over time.

The screenshot shows the SumoLogic Status interface. At the top, there's a search bar and a dashboard menu. Below that is a chart titled "Total Message Volume" showing a single data point for 3/07/2014 at 10:56:00 AM with a value of 200. To the right, a dropdown menu for "Manage" is open, with "Status" highlighted. Other options in the dropdown include Account, Collectors, Users, and Security.

- In the SumoLogic dashboard, click **Search**. Place the cursor in the search box, then press **Enter** to submit an empty search query.

The screenshot shows the SumoLogic search results page. At the top, it says "Status: Done gathering results ELAPSED TIME: 00:00:01 RESULTS: 167 SESSION: 070720140304109". Below this is a histogram showing message volume over time. The main area displays a list of log messages from host 99.74.215.164. The log entries are as follows:

#	Time	Message
1	03/07/2014 11:48:34 AM	203 <14>1 2014-03-07T11:48:34+00:00 loggregator 0f65cbbb-aabb-49a8-8073-f604fcc19260 [App/0] -- - 17:48:34,418 INFO DispatcherServlet:480 - FrameworkServlet 'appServlet': initialization completed in 1287 ms Host: 99.74.215.164 ▾ Name: Http Input ▾ Category: Http Input ▾
2	03/07/2014 11:48:34 AM	143 <11>1 2014-03-07T11:48:34+00:00 loggregator 0f65cbbb-aabb-49a8-8073-f604fcc19260 [App/0] -- - INFO: Starting ProtocolHandler ["http-bio-61215"] Host: 99.74.215.164 ▾ Name: Http Input ▾ Category: Http Input ▾
3	03/07/2014 11:48:34 AM	160 <11>1 2014-03-07T11:48:34+00:00 loggregator 0f65cbbb-aabb-49a8-8073-f604fcc19260 [App/0] -- - Mar 07, 2014 5:48:34 PM org.apache.catalina.startup.Catalina start Host: 99.74.215.164 ▾ Name: Http Input ▾ Category: Http Input ▾
4	03/07/2014 11:48:34 AM	128 <11>1 2014-03-07T11:48:34+00:00 loggregator 0f65cbbb-aabb-49a8-8073-f604fcc19260 [App/0] -- - INFO: Server startup in 24233 ms Host: 99.74.215.164 ▾ Name: Http Input ▾ Category: Http Input ▾
5	03/07/2014 11:48:34 AM	158 <11>1 2014-03-07T11:48:34+00:00 loggregator 0f65cbbb-aabb-49a8-8073-f604fcc19260 [App/0] -- - Mar 07, 2014 5:48:34 PM org.apache.coyote.AbstractProtocol start Host: 99.74.215.164 ▾ Name: Http Input ▾ Category: Http Input ▾
6	03/07/2014 11:48:33 AM	162 <11>1 2014-03-07T11:48:33+00:00 loggregator 0f65cbbb-aabb-49a8-8073-f604fcc19260 [App/0] -- - Mar 07, 2014 5:48:33 PM org.apache.catalina.util.Sessiondecorator createSecureRandom Host: 99.74.215.164 ▾ Name: Http Input ▾ Category: Http Input ▾
7	03/07/2014 11:48:33 AM	195 <11>1 2014-03-07T11:48:33+00:00 loggregator 0f65cbbb-aabb-49a8-8073-f604fcc19260 [App/0] -- - Mar 07, 2014 5:48:33 PM auto_reconfig.org.springframework.cloud.AbstractCloudConnector getServiceInfo Host: 99.74.215.164 ▾ Name: Http Input ▾ Category: Http Input ▾
8	03/07/2014 11:48:33 AM	238 <14>1 2014-03-07T11:48:33+00:00 loggregator 0f65cbbb-aabb-49a8-8073-f604fcc19260 [App/0] -- - 17:48:33,286 INFO XmlBeanDefinitionReader:416 - Loading XML bean definitions from class path resource [META-INF/cloud/cloudfoundry-auto-reconfiguration-context.xml] Host: 99.74.215.164 ▾ Name: Http Input ▾ Category: Http Input ▾
9	03/07/2014 11:48:33 AM	367 <14>1 2014-03-07T11:48:33+00:00 loggregator 0f65cbbb-aabb-49a8-8073-f604fcc19260 [App/0] -- - 17:48:33,907 INFO RequestMappingHandlerMapping:181 - Mapped {"{/info},methods=[],params=[],headers[],consumes[],produces[],custom=[]}" onto public org.cloudfoundry.samples.music.domain.ApplicationInfo org.cloudfoundry.samples.music.web.controllers.InfoController.info() Host: 99.74.215.164 ▾ Name: Http Input ▾ Category: Http Input ▾

Logsene

Note: Logsene uses HTTPS for communication. HTTPS is supported in Cloud Foundry v158 and above.

From your Sematext account:

- Click the [Create App / Logsene App](#) menu item. Enter a name and click the **Add Application** button to create the Logsene App.
- Create the log drain service in Cloud Foundry using the displayed URL.

```
$ cf cups logsene-log-drain -l https://logsene-cf-receiver.sematest.com/YOUR_LOGSENE_TOKEN
```

- 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
```

- 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 in order 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.

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 app is bound to the service and was restarted after binding
- The service port number matches the TCP port number in Splunk

Streaming Application Logs with Fluentd

Page last updated:

Fluentd [\[x\]](#) 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](#) [\[x\]](#) 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](#) [\[x\]](#) 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> [\[x\]](#)

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": "jdbc:mysql://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` 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 in a file. This parameter specifies the `gid` and `uid` to use when mounting the share to the app.

The following example shows binding `my-app` to the `nfs_service_instance`, passing an in-line JSON string:

```
cf bind-service my-app nfs_service_instance -c '{"uid": "1000", "gid": "1000"}'
```

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
<code>container_dir</code>	String containing the path to the mounted volume that you bound to your app.
<code>device_type</code>	The NFS volume release. This currently only supports <code>shared</code> devices. A <code>shared</code> device represents a distributed file system that can mount on all app instances simultaneously.
<code>mode</code>	String that informs what type of access your app has to NFS, either read-only, <code>ro</code> , or read and write, <code>rw</code> .

Security Guide

This guide is intended for administrators, developers, and anyone interested in making their Pivotal Cloud Foundry (PCF) deployment more secure. Use this guide to learn how PCF manages infrastructure security, roles and permissions, container hardening, and more.

To learn about Pivotal's vulnerability reporting and responsible disclosure process, read the [PCF Security Overview and Policy](#). You can review the latest CVEs on the [Pivotal Application Security page](#).

If you are new to PCF security, start with the [Concepts](#) section.

Security Concepts

Start here if you need a high-level view of how PCF manages security.

- [Security Concepts](#)

PCF Security Processes

This section provides high-level content on how Pivotal remediates, discloses, and prevents security events.

- [PCF Security Processes](#)

Identity Management

This section includes topics on identity, authorization, authentication, and access management.

- [Identity Management](#)

PCF Infrastructure Security

This section includes topics on securing connections and components, such as the Ops Manager Director and stemcells, inside a PCF deployment.

- [PCF Infrastructure Security](#)

Network Security

This section includes content about setting up and securing network connections.

- [Network Security](#)

Security-Related PCF Tiles

This section includes content about PCF-related services that can enhance security in a deployment.

- [Security-Related PCF Tiles](#)

Other Security Topics

This section includes content that may help you manage other security concerns.

- [Other Security Topics](#)

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.

PCF Security Processes

This section introduces some of Pivotal's security-related processes.

- [Pivotal Cloud Foundry Security Overview and Policy](#)

This topic explains Pivotal's responsible disclosure and vulnerability response procedures.

- [PCF Testing, Release, and Security Lifecycle](#)

This topic explains how Pivotal's practices, tools, and organizational structures work together to create and support stable releases of Pivotal Cloud Foundry (PCF).

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 releases of software in the PCF Suite to address Low / Medium severity vulnerability exploits. These patch releases take place during the first week each month. 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 [Elastic Runtime](#).

[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

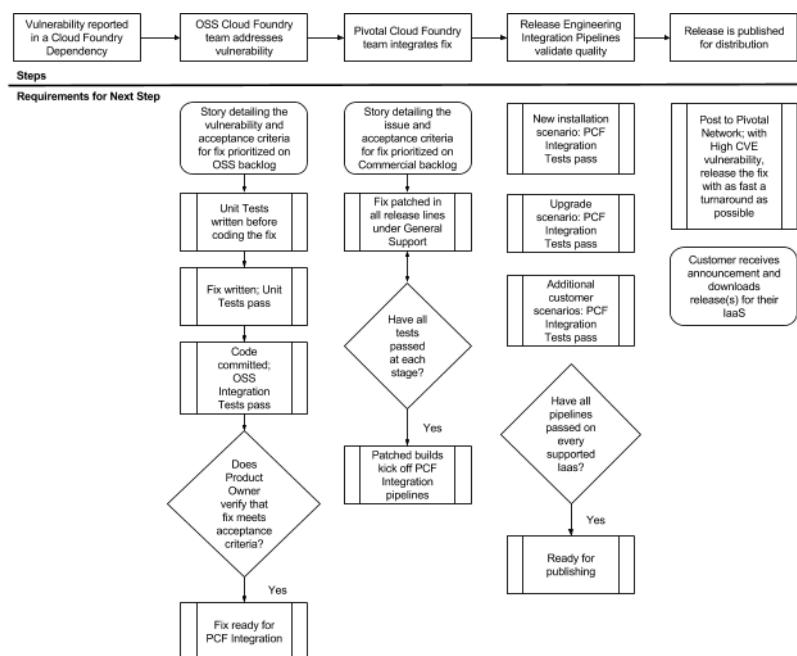
Pivotal also monitors externally-reported vulnerabilities from many sources, including:

- 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 \(UAAC\) !\[\]\(529302382f60930c16d437445458dba0_img.jpg\)](#)
- [Security Event Logging for Cloud Controller and UAA !\[\]\(694fd1d4c1e8b0e89b817477a14234c7_img.jpg\)](#)

PCF Infrastructure Security

This section provides links to topics about infrastructure hardening and security:

- [Rotating Non-Configurable TLS/SSL Certificates](#)

This topic explains how to use an API endpoint to rotate certain types of certificates.

- [Stemcell Hardening FAQ](#)

This topic provides an overview of how PCF stemcells stay secure.

Regenerating and Rotating Non-Configurable TLS/SSL Certificates

Depending on the requirements of your deployment, at some point you may need to rotate your CA certificates. Certificates can expire or fall out of currency, or your organization's security compliance policies may require you to rotate certificates periodically.

Rotate the certificates in your Pivotal Cloud Foundry (PCF) deploying using API calls in the command line. PCF provides different API calls with which to manage certificates and certificate authorities (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 Ops Manager Director then passes the certificates to other components in your PCF deployment.

Follow the procedures below in order to apply new CAs with minimal risk.

 **Note:** These procedures require you to return to Ops Manager and click **Apply Changes** periodically. Clicking **Apply Changes** redeloys 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.

Creating a New Certificate Authority (CA)

1. Open the command line.
2. Open a web browser and navigate to Ops Manager.
3. In Ops Manager, click **Apply Changes**.
4. On the command line, enter the following API call with an empty request:

```
curl "https://EXAMPLE.com/api/v0/certificateAuthorities/EXAMPLE-CERT-GUID/activate" \
-X POST \
-H "Authorization: Bearer YOUR-UAA-ACCESS-TOKEN" \
-H "Content-Type: application/json" \
-d '{}'
```

The API returns a successful response, including a new certificate.

```
HTTP/1.1 200 OK
{
  "guid": "f7be18f34f2a7a9403c3",
  "issuer": "Pivotal",
  "created_on": "2017-01-19",
  "expires_on": "2021-01-19",
  "active": false,
  "cert_pem": "-----BEGIN EXAMPLE CERTIFICATE-----\nMIIC+zCCAEoAwIBAgIBADANBgkqhkiG9w0BAQsFADAfMQswCQYDVQQGEwJVUzEQ\nMA4GA1UECgwHUGI2b3RhDAeFw0xNzAxMTgyMTQyMjVaFw0yMTAxMTkyMTQyMjVa\nMB8xCzAJBgNVBAYTMRAwDgYDVQQDAdqaXzvDFsMIIBjANBgkqhkiG9w0B\nAQEFAAOCAQ8AMhIBCgKCAQEAYV4OhPIIZTeym9OcdCNvip9Ev0jPPLo9WPLOUzT\nIrpDx3nG/TgD+DP09mwVXfqwBljmoj9DqRED1x/6bc0Ki/BaFo/P4MmOKm3QnDCt\no+4RUvLkQqgA++2HYRNTKWJ5fXmERs8IK9AXXT7RKXhktWWU3oNGf7zo0e3YKw\n107dIWTh1NwlbNgCT1AurIDssyOZy1HVzLDPtUR2MxjhlmSCLsOw3qUDQjatjXKw\n82RjcrswjG3nV2hvD4/aTOiHuKM3+AGbmS2MdiOvFOh/Y79tUp89csK0gs6uOd\nmyfdxzDihe4DcKw5CzUTfHNKXgHyeoVOBPeVQTp4Up1QIDAQABo0lwQDAdBgNV\nHQ4EFgQoUYh4y7vEuImLSxM0CKR8uVqxX/gwDwYDVROTAQH/BAUwAwEB/zAOBgNV\nHQ8BAf8EBAMCAQYwDQYJKoZIhvcNAQELBQADggEBALmHOPxdyBGmuR0HgR9V4TwJ\ntnKFdFFQIGLKVT7am5z6G20q5cwACFHWAffrPG4W9Jm577QtewiY/Rad/PbkY0YSY\nrehLThKdkrfNjxjxI0H2s7qlBFjJ0wBZHvVmDsO6A9PfkAp4eJqvRMuL/xGmSQ\niVkgzYmn7FgHyFbd9D9X5Yw8fWGSeVBPPikcONdRvjw9aEeAtbGEh8eZCP\naBQOgsx7b33RuR+CTNqThXY9k8d7/7ba4KVdd4gP8ynFgvvnDQOjeJZ6Go5QY5HA\nR+Oglzs3PFW8pAYcvWrXKR0rE8fL5o9qgTyjmO+5yyyWITyrKPqqIUIvMCdNr84=\n-----END EXAMPLE CERTIFICATE-----\n"
}
```

This creates a new CA.

Activating the New CA

1. On the command line, enter the following API call:

```
curl "https://EXAMPLE.com/api/v0/certificateAuthorities" \
-X GET \
-H "Authorization: Bearer YOUR-UAA-ACCESS-TOKEN"
```

The new CA displays, marked as inactive.

2. In Ops Manager, click **Apply Changes**.

3. On the command line, enter the following API call with an empty request:

```
curl "https://EXAMPLE.com/api/v0/certificateAuthorities/EXAMPLE-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
```

This activates the new CA.

Regenerating Non-Configurable Certificates to Apply the New CA

1. On the command line, enter the following API call with an empty request:

```
curl "https://EXAMPLE.com/api/v0/certificateAuthorities/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
```

This regenerates all non-configurable certificates and applies the new CA to your existing Ops Manager Director.

2. In Ops Manager, click **Apply Changes**.

Deleting the Old CA

1. On the command line, enter the following API call:

```
curl "https://EXAMPLE.com/api/v0/certificateAuthorities/:guid" \
-X DELETE \
-H "Authorization: Bearer YOUR-UAA-ACCESS-TOKEN"
```

The API returns a successful response.

```
HTTP/1.1 200 OK
```

This deletes the old, inactive CA.

2. In Ops Manager, click **Apply Changes**.

Stemcell Hardening FAQ

 **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. The rsyslog package is installed and activated as a BOSH add-on.
- ix. The rsyslog log is configured for proper ownership and permission mask (via a BOSH Add-on).
- x. The rsyslog utility is configured to send logs to a remote log host for storage (via a BOSH Add-on).
- xi. 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](#)

TLS Connections in PCF Deployments

Pivotal Cloud Foundry (PCF) uses Transport Layer Security (TLS) protocols to secure connections between internal components 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.

By default, PCF uses a limited set of cipher suites and TLS versions to secure connections.

The TLS versions and cipher suites PCF supports are in the table below.

PCF version	TLS version	Supported cipher suites
1.10	1.2	<ul style="list-style-type: none">• 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

Security-Related PCF Tiles

This section provides information on 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 your applications. Buildpacks typically examine user-provided artifacts to determine what dependencies to download and how to configure applications to communicate with bound services.

When you push an application, Cloud Foundry automatically detects which buildpack is required and installs it on the Diego cell where the application needs to run.

 **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](#) application.

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
Node.js	Node or JavaScript	Node.js source
.NET Core	.NET Core	.NET Core 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](#).

Using, Developing, and Customizing Buildpacks

For information about using buildpacks, see [Using Buildpacks](#).

For information about customizing existing buildpacks and developing new buildpacks, see [Developing Buildpacks](#).

For information about updating and releasing a new version of a Cloud Foundry (CF) buildpack through the CF Buildpacks Team Concourse pipeline, see [CF Buildpack Team CI](#). You can use this as a model for using Concourse to build and release new versions of your own

buildpacks.

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

Certificates deployed through [BOSH configured custom trusted certificates](#) exist in 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 1.5

- [Sample Go 1.5 app](#)

An example `Godeps/Godeps.json`:

```
{  
  "ImportPath": "go_app",  
  "GoVersion": "go1.5",  
  "Deps": []  
}
```

An example `manifest.yml`:

```
--  
applications:  
  - name: my-app-name
```

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: app-package-name
```

Go 1.5

 **NOTE:** For Go 1.5, native vendoring is disabled by default. You must set the `GO15VENDOREXPERIMENT` environment variable to `1` in your `manifest.yml` file to use this native vendoring.

If you use the `vendor/` directory for dependencies, you can set the Go version with the `GOVERSION` environment variable.

An example `manifest.yml` file:

```
---  
applications:  
- name: my-app-name  
  env:  
    GOVERSION: go1.5  
    GOPACKAGENAME: app-package-name  
    GO15VENDOREXPERIMENT: 1
```

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: 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: 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` and supports [BOSH configured custom trusted certificates](#) with no additional configuration necessary.

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

Page last updated:

Use the Java buildpack with applications written in Grails, Play, Spring, or any other JVM-based language or framework.

See the following topics:

- [Getting Started Deploying Grails Apps](#)
- [Getting Started Deploying Ratpack Apps](#)
- [Getting Started Deploying Spring Apps](#)
- [Tips for Java Developers](#)
- [Configuring Service Connections for Grails](#)
- [Configuring Service Connections for Play](#)
- [Configuring Service Connections for Spring](#)
- [Cloud Foundry Eclipse Plugin](#)
- [Cloud Foundry Java Client Library](#)
- [BOSH Configured Custom Trusted Certificate Support](#)

See the [Java Buildpack Release Notes](#) for information about specific versions. You can find the source for the Java buildpack on GitHub: <https://github.com/cloudfoundry/java-buildpack>

Buildpack Logging and Application Logging

The 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.

Getting Started Deploying Grails Apps

Page last updated:

This guide is intended to walk you through deploying a Grails app to Elastic Runtime. 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 Elastic Runtime.

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	<code>build.gradle</code>	Gradle User Guide
Grails	<code>BuildConfig.groovy</code>	Grails: Configuration - Reference Documentation
Maven	<code>pom.xml</code>	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 the [Apps Manager](#) to perform this task.

Elastic Runtime supports two types of service instances:

- Managed services integrate with Elastic Runtime through service brokers that offer services and plans and manage the service calls between Elastic Runtime 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, Elastic Runtime 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 Elastic Runtime 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 Elastic Runtime 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 `./grails 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 (0.0s)
----> 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 Elastic Runtime!

Use the cf CLI or the [Apps Manager](#) to review information and administer your app and your Elastic Runtime 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 response.

Created

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 v6](#) topic.

 **Note:** You cannot perform certain tasks in the CLI or the [Apps Manager](#) because these are commands that only a Elastic Runtime administrator can run. If you are not a Elastic Runtime 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 Elastic Runtime. 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 the [Apps Manager](#), or you might have to delete the app and redeploy.

App Requires a Unique URL

Elastic Runtime requires that each app that you deploy have a unique URL. Otherwise, the new app URL collides with an existing app URL and Elastic Runtime 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 Elastic Runtime. 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 Elastic Runtime.

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	<code>build.gradle</code>	Gradle User Guide
Maven	<code>pom.xml</code>	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 the [Apps Manager](#) to perform this task.

Elastic Runtime supports two types of service instances:

- Managed services integrate with Elastic Runtime through service brokers that offer services and plans and manage the service calls between Elastic Runtime 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, Elastic Runtime 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 Elastic Runtime 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 Elastic Runtime 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 cloudunder@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 cloudunder@example.com...
OK

Starting app pong_matcher_groovy in org Cloud-Apps / space development as cloudunder@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 cloudunder@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 Elastic Runtime!

Use the cf CLI or the [Apps Manager](#) to review information and administer your app and your Elastic Runtime 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 v6](#) topic.

 **Note:** You cannot perform certain tasks in the CLI or the [Apps Manager](#) because these are commands that only a Elastic Runtime administrator can run. If you are not a Elastic Runtime 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 Elastic Runtime. 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 the [Apps Manager](#), or you might have to delete the app and redeploy.

App Requires a Unique URL

Elastic Runtime requires that each app that you deploy have a unique URL. Otherwise, the new app URL collides with an existing app URL and Elastic Runtime 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 Elastic Runtime. You can choose whether to push a sample app, your own app, or both.

If at any time you experience a problem following the steps below, try checking the [Known Issues](#) topic, or refer to the [Troubleshooting Application Deployment and Health](#) topic for more tips.

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 Elastic Runtime.

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	<code>build.gradle</code>	Gradle User Guide
Maven	<code>pom.xml</code>	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>
```

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

Elastic Runtime 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 Elastic Runtime 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 Elastic Runtime 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 my-spring-
HOSTNAME`

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 Elastic Runtime!

Use the cf CLI or the [Apps Manager](#) to review information and administer your app and your Elastic Runtime 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.

Elastic Runtime 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 v6](#) topic.

 **Note:** You cannot perform certain tasks in the CLI or the [Apps Manager](#) because these are commands that only a Elastic Runtime administrator can run. If you are not a Elastic Runtime 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 Elastic Runtime. 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 the [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, Elastic Runtime 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

Elastic Runtime requires that each app that you deploy have a unique URL. Otherwise, the new app URL collides with an existing app URL and Elastic Runtime 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.

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 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 Buildpack

For detailed information about using, configuring, and extending the Cloud Foundry Java buildpack, see <https://github.com/cloudfoundry/java-buildpack>.

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#0928916a2dd78e9fa9469c558046cef09f60e5d  
--> 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 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 Elastic Runtime 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 [here](#) 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: Metaspase
```

- **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 `metaspase`. 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`, `metaspase`, 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 [X](#) 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 X](#) (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 X](#) 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.

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 re-configures the plugin with connection information when your app starts.

If you were using 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 overridden 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
<code>cloud.application.name</code>	The name provided when the application was pushed to Cloud Foundry.
<code>cloud.provider.url</code>	The URL of the cloud hosting the application, such as <code>cloudfoundry.com</code> .

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**.

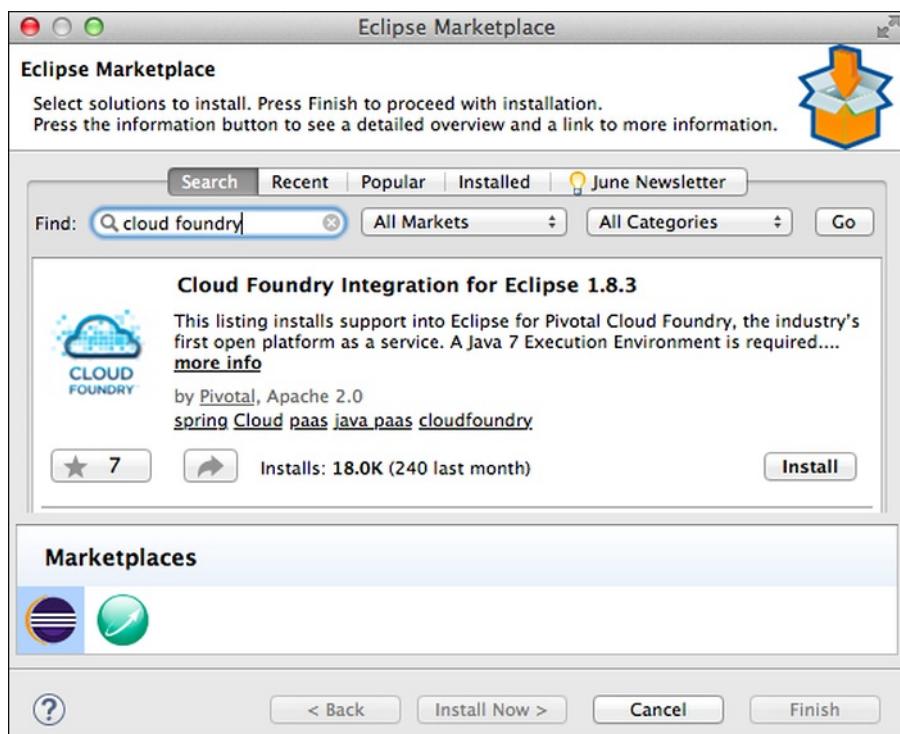
Follow the installation instructions appropriate for your environment:

- [Install to Eclipse from Marketplace](#)
- [Install to STS from IDE Extensions Tab](#)
- [Install from a Local Repository](#)

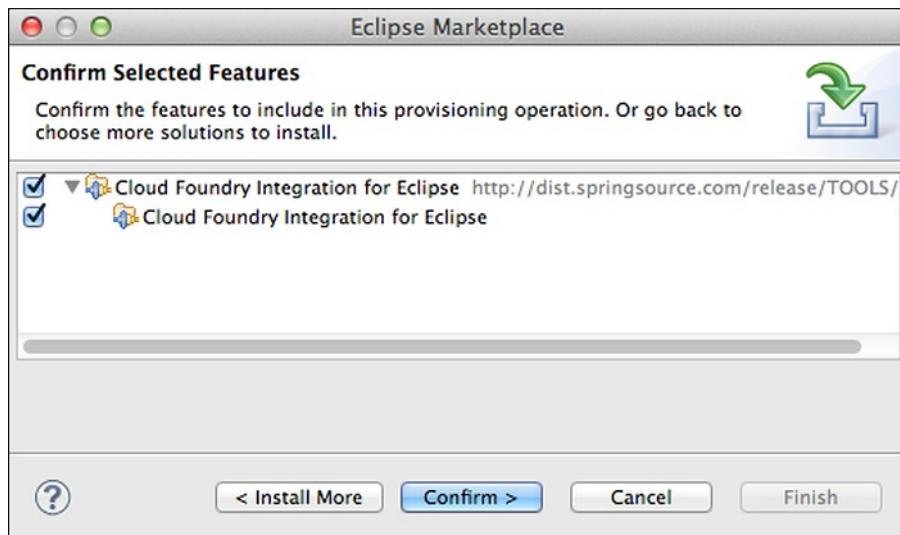
Install to Eclipse from Marketplace

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**.



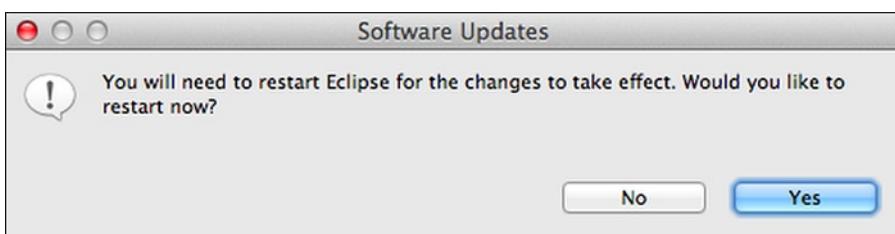
5. In the **Confirm Selected Features** window, click **Confirm**.



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.



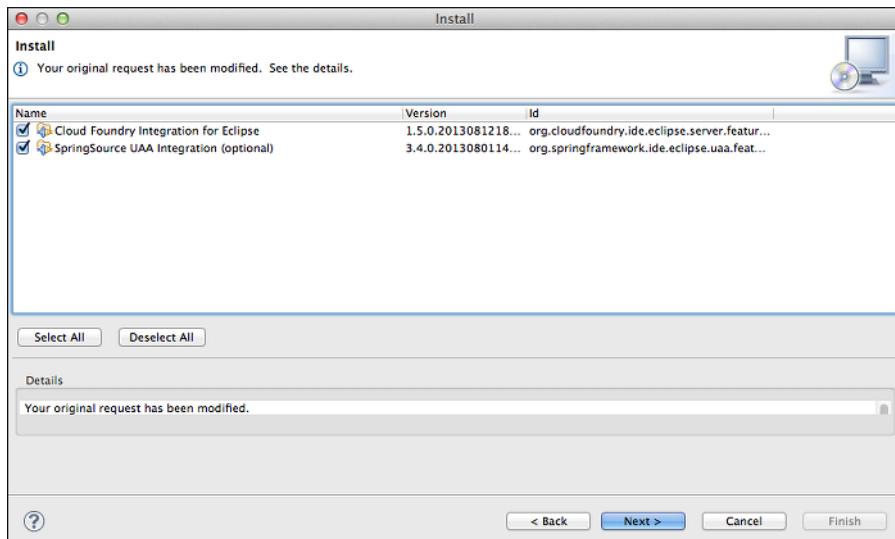
Install to STS from IDE Extensions Tab

Follow these instructions to install the Cloud Foundry Eclipse Plugin to Spring Tool Suite (STS) from the **IDE Extensions** tab.

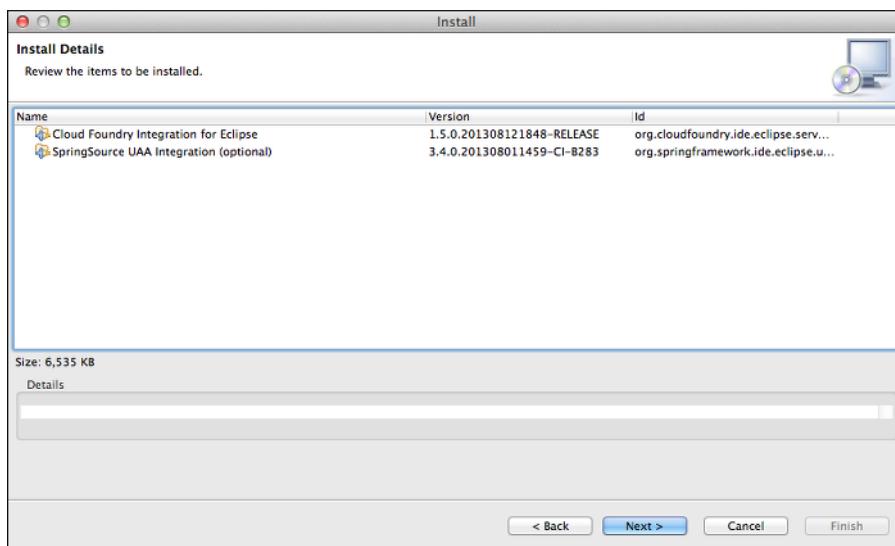
1. Start STS.
2. On the STS Dashboard, click **IDE Extensions**.

3. Enter "Cloud Foundry" in the **Find** field.
4. Select **Cloud Foundry Integration for Eclipse** and click **Install**.

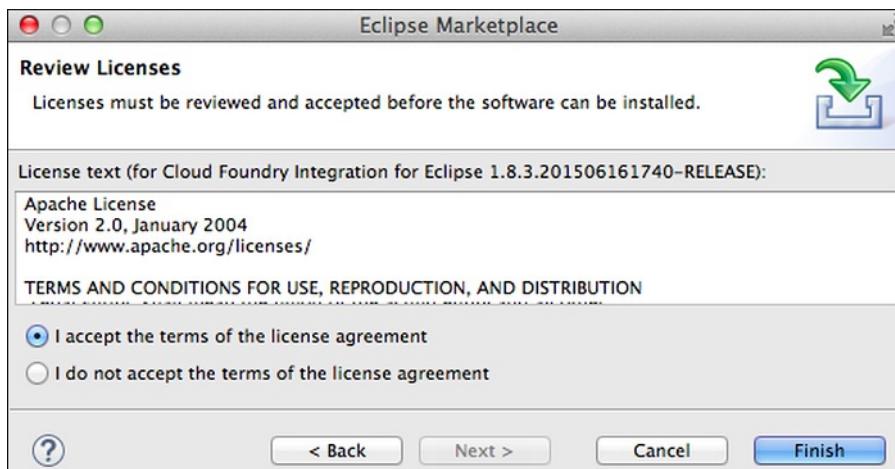
5. In the **Install** window, click **Next**.



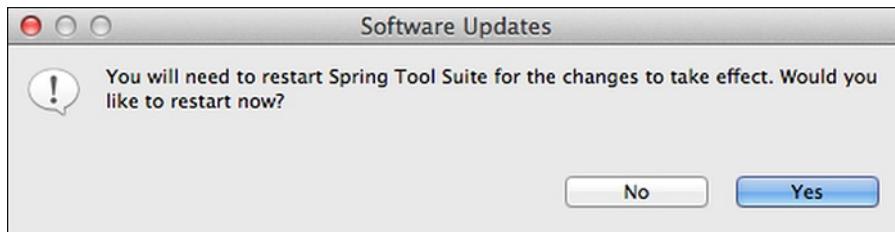
- In the **Install Details** window, click **Next**.



- The **Review Licenses** window appears. Select "I accept the terms of the license agreement" and click **Finish**.



- 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**.

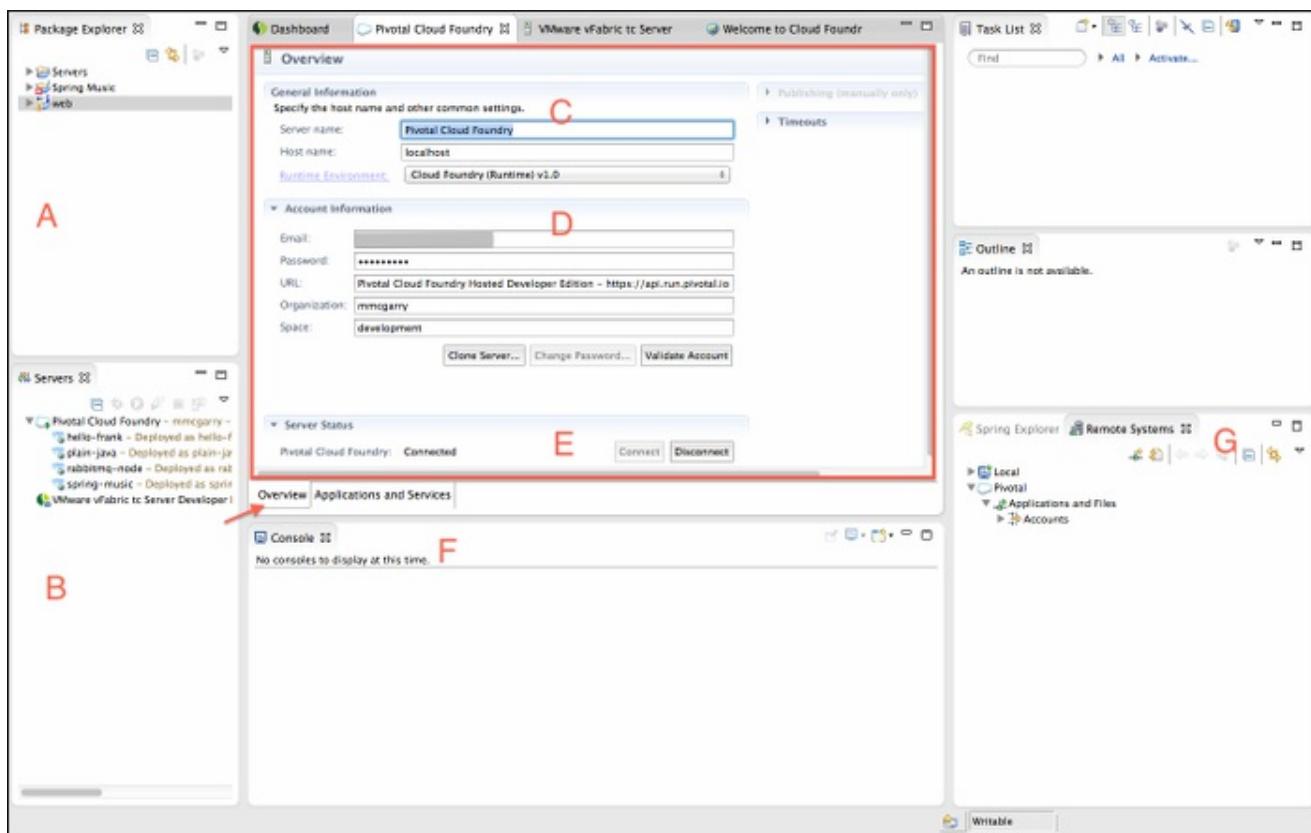
The Cloud Foundry editor, outlined in red in the screenshot below, is the primary plugin user interface. Some workflows involve interacting with standard elements of the Eclipse user interface, such as the **Project Explorer** and the **Console** and **Servers** views.

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 following panes and views are present when the **Overview** tab is selected:

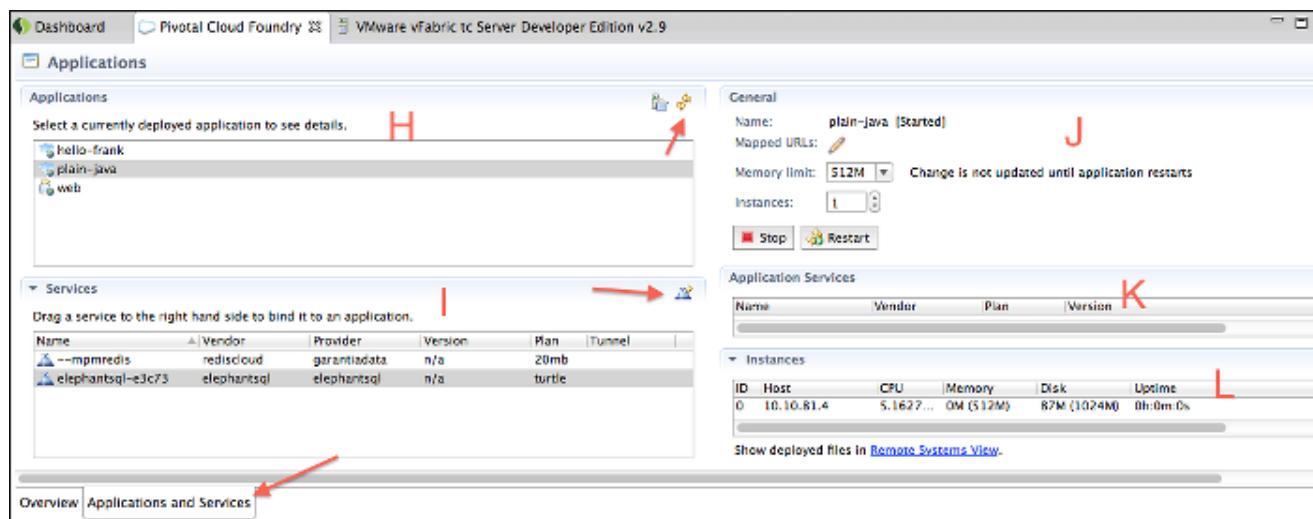
- A – The **Package Explorer** view lists the projects in the current workspace.
- B – The **Servers** view lists server instances configured in the current workspace. A server of type **Pivotal Cloud Foundry** represents a targeted space in a Cloud Foundry instance.
- C – The **General Information** 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 following 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](#).

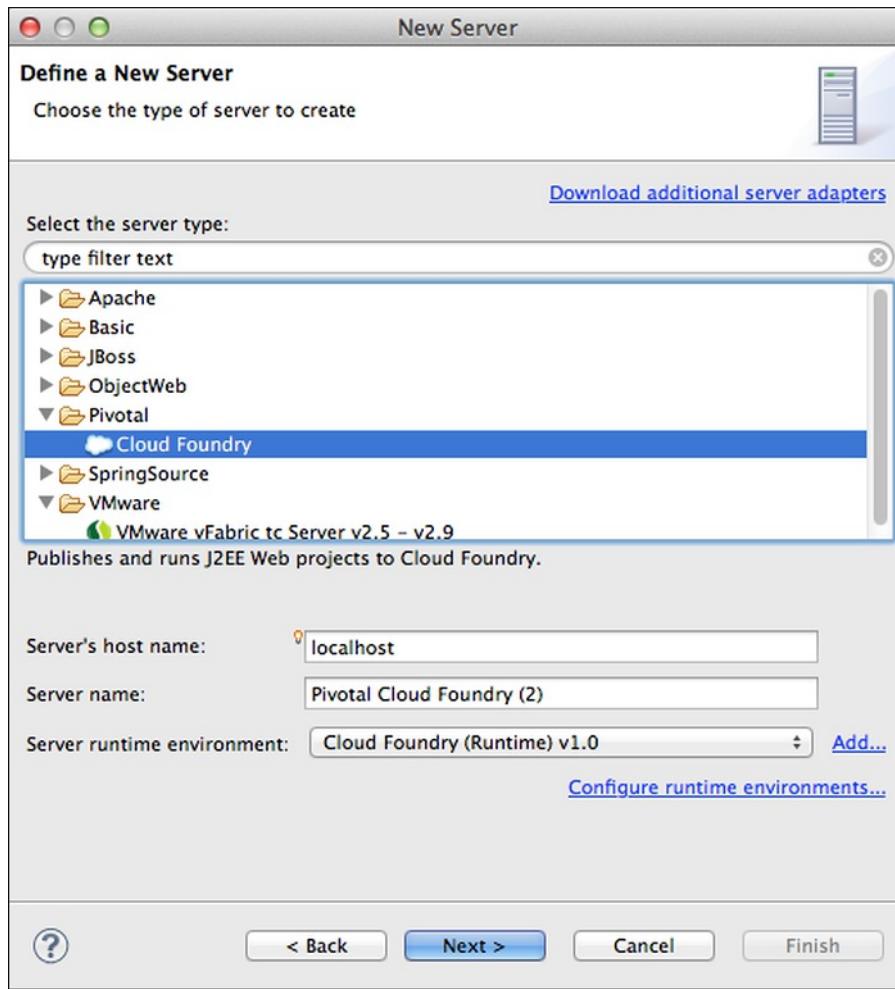


Create a Cloud Foundry Server

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

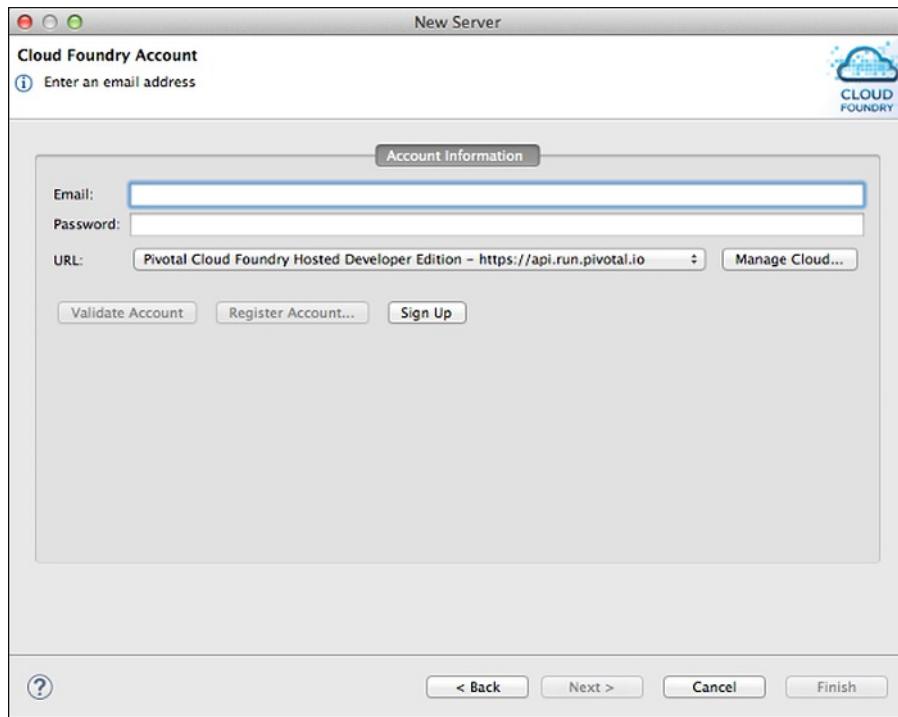


3. In the **Cloud Foundry Account** window, if you already have a Pivotal Cloud Foundry Hosted Developer Edition account, enter your email account and password credentials and click **Validate Account**.

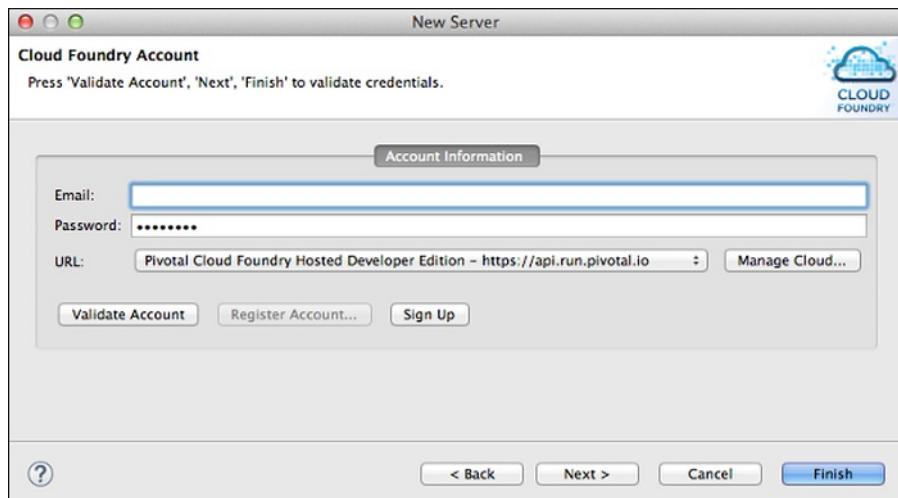
Note: By default, the **URL** field points to the Pivotal Cloud Foundry Hosted Developer Edition URL, `https://api.run.pivotal.io`. If you have a Pivotal Elastic Runtime account, refer to the [Logging in to Apps Manager](#) topic to determine your Pivotal Elastic Runtime URL. Click **Manage Cloud...** 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.

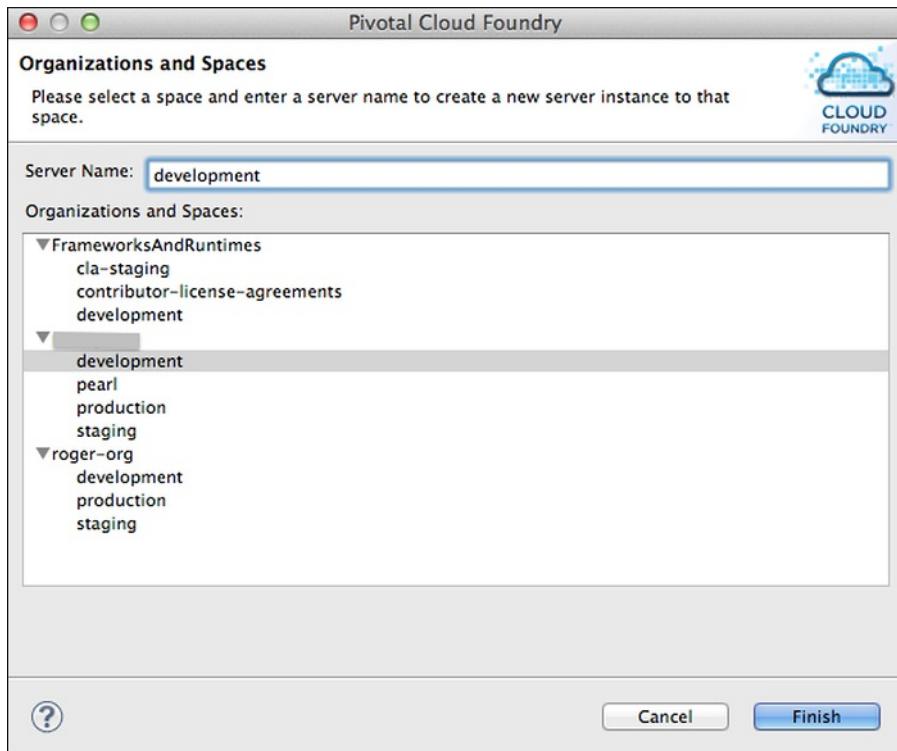


4. The **Cloud Foundry Account** window is refreshed and displays a message indicating whether or not your credentials were valid. Click **Next**.



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.



- 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:

- 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.
- 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.

Application details

Specify application details.

Name:

Buildpack URL (optional):

< Back Next > Cancel **Finish**

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.

Launch deployment

Specify the deployment details



Host:

Domain:

Deployed URL:

Memory Reservation:

Start application on deployment

< Back Next > Cancel **Finish**

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](#).

Name	Vendor	Provider	Version	Plan	Tunnel
<input checked="" type="checkbox"/> --mpmredis	rediscloud	garantiadata	n/a	20mb	
<input checked="" type="checkbox"/> elephantsql-e3c73	elephantsql	elephantsql	n/a	turtle	

Buttons at the bottom: ? < Back Next > Cancel Finish

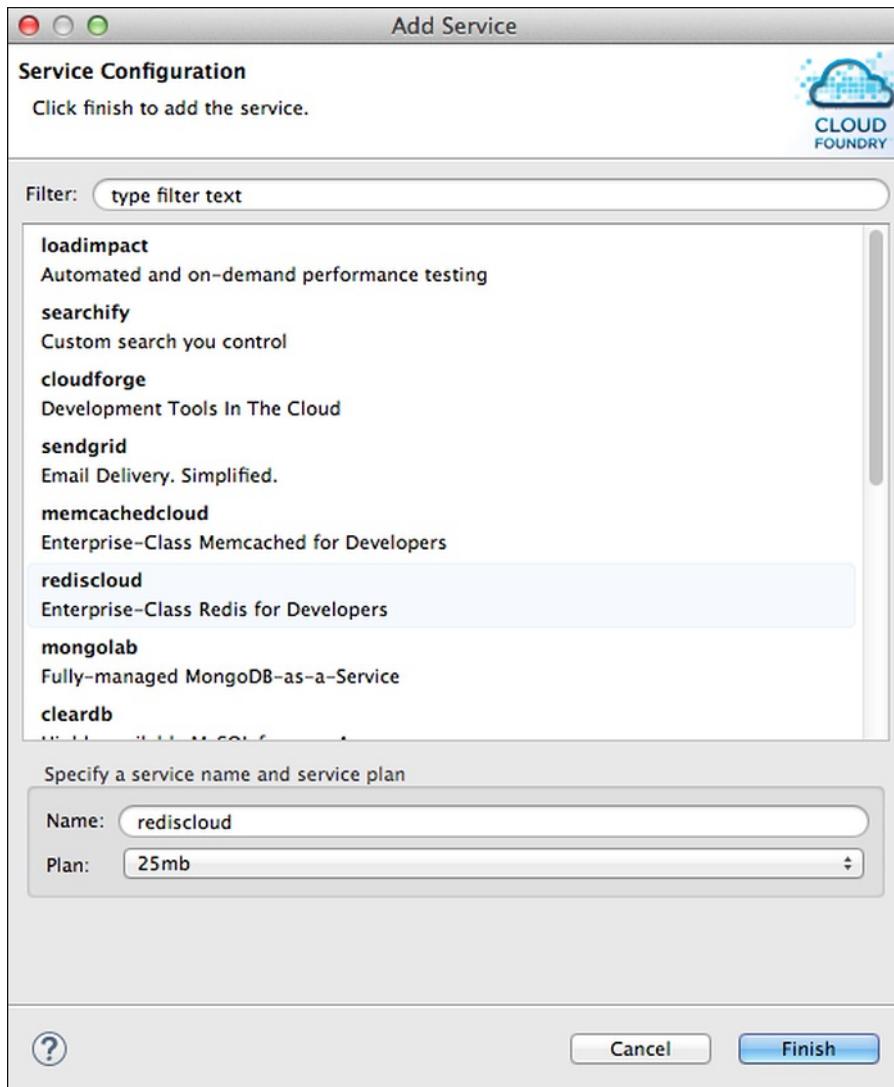
As the deployment proceeds, progress messages appear in the **Console** view. When deployment is complete, the application is listed in the **Applications** pane.

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

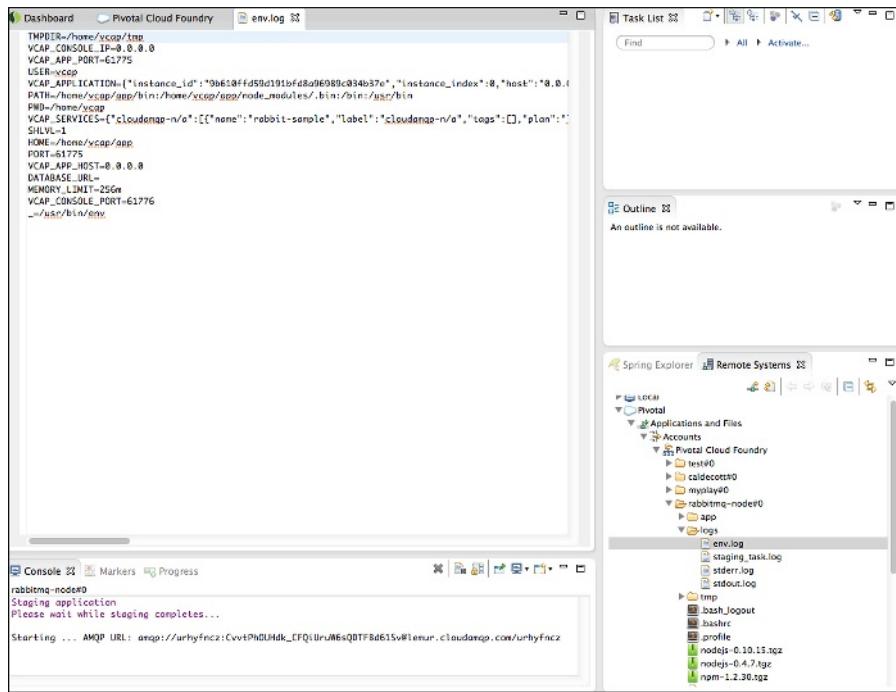
You can bind a service to an application when you deploy it. To bind a service to an application that is already deployed, drag the service from the **Services** pane to the **Application Services** pane. (See the area labelled “G” in the screenshot in the [Applications and Services](#) above.)

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, the application statistics can be manually refreshed by clicking the **Refresh** button in the top, right corner of the “Applications” pane, labelled “H” in the screenshot in [Applications and Services Tab](#).

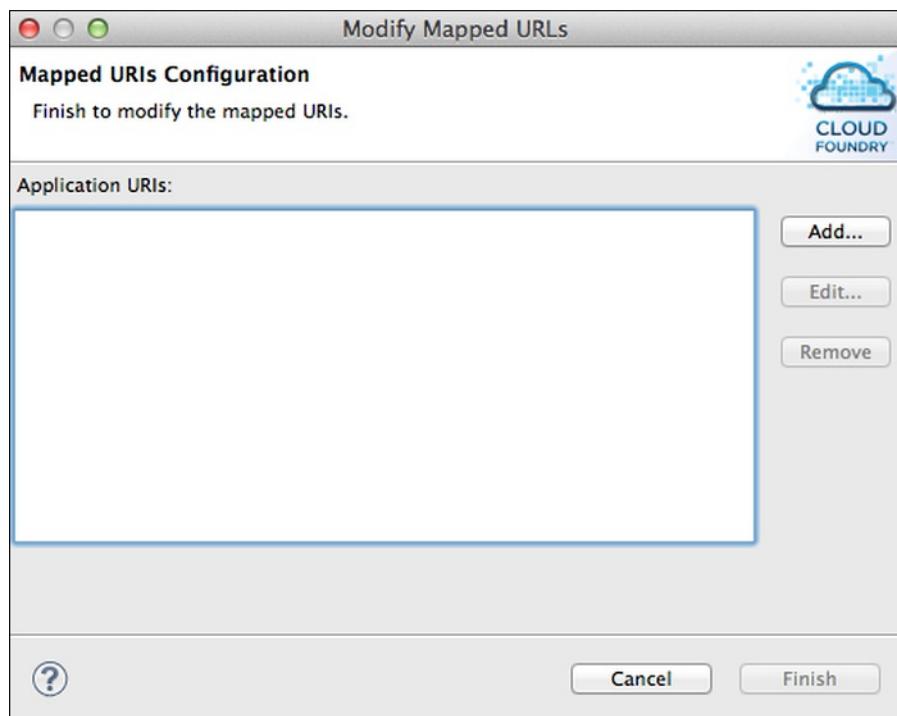
Push Application Changes

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 URLs 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.

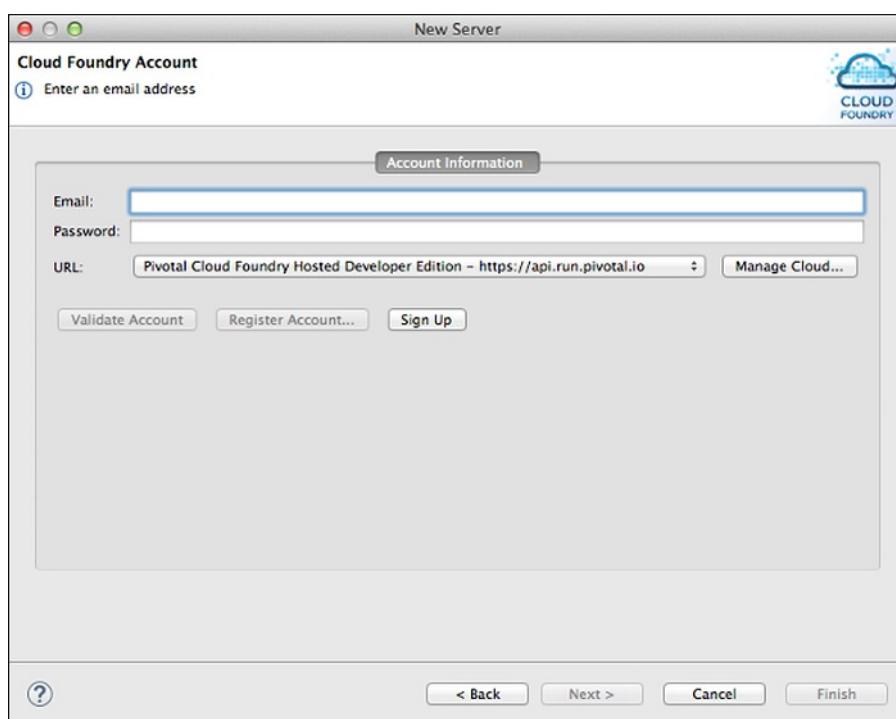
Follow the step below to clone a server:

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**.

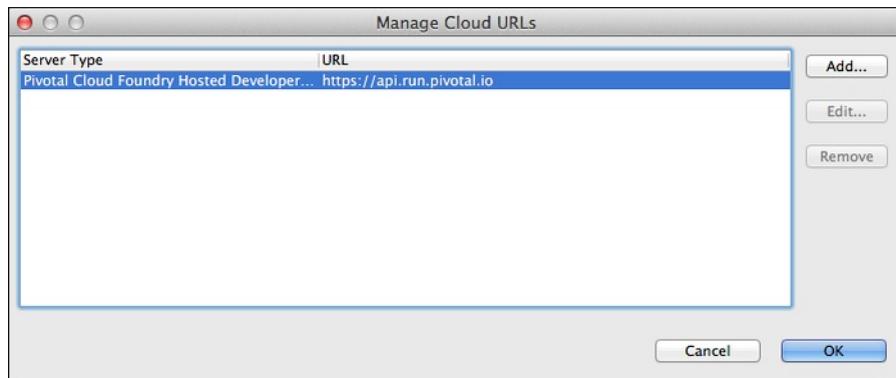
Add a Cloud Foundry Instance URL

You can configure the plugin to work with any Cloud Foundry instances to which you have access. To do so:

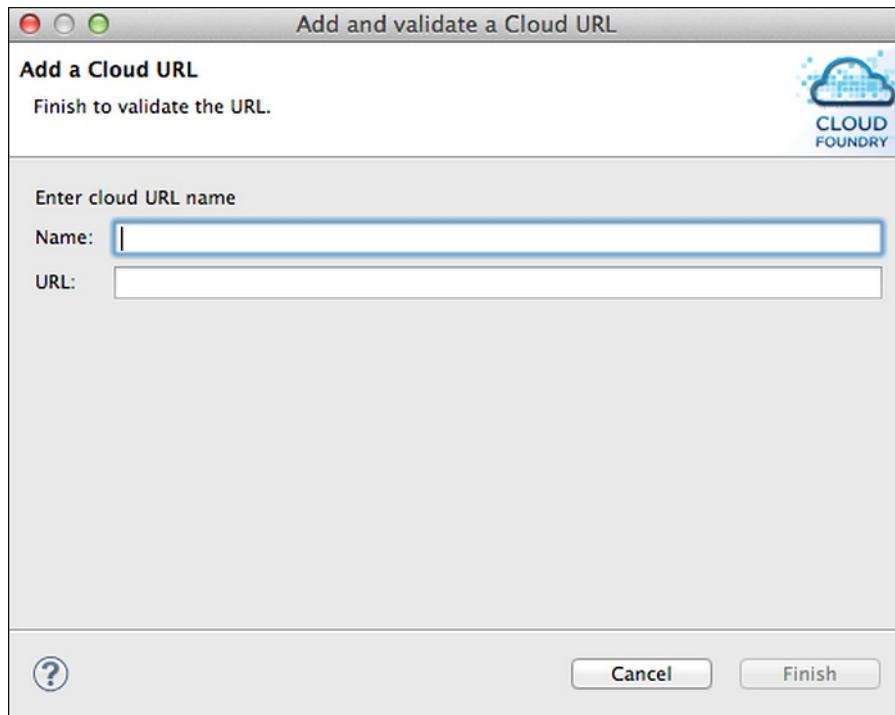
1. Perform steps 1 and 2 of [Create a Cloud Foundry Server](#).
2. In the **Cloud Foundry Account** window, enter the email account and password that you use to log on to the target instance, then click **Manage Cloud URLs**



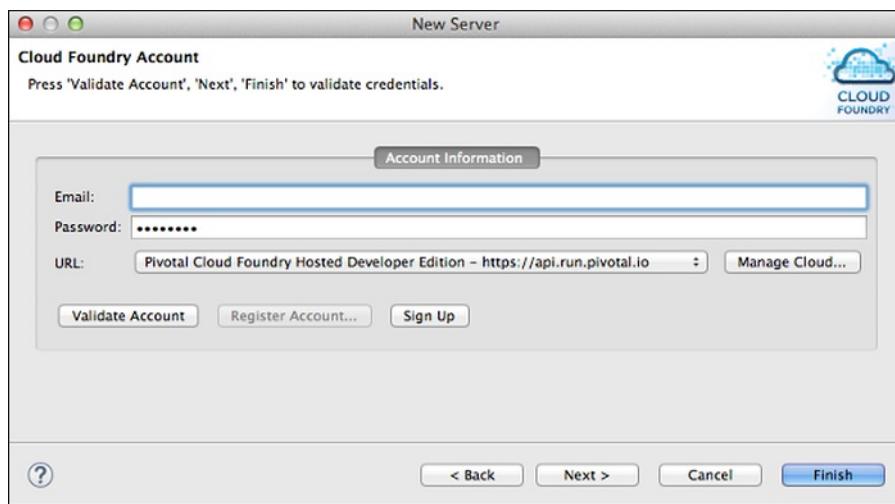
3. In the **Manage Cloud URLs** window, click **Add**.



4. In the **Add a Cloud URL** window, enter the name and URL of the target cloud instance and click **Finish**.

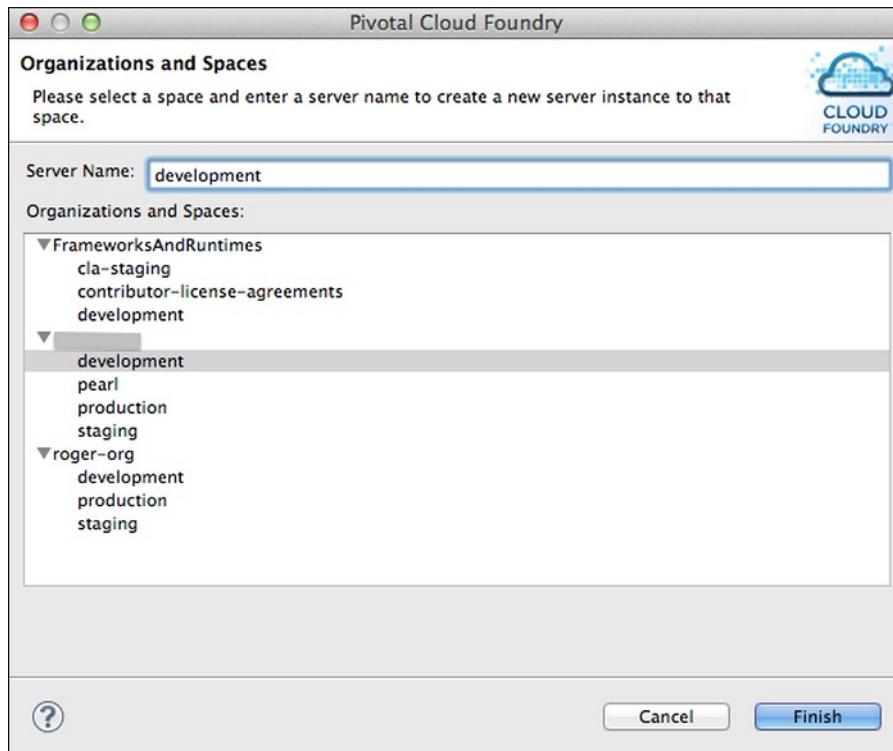


5. The new cloud instance should appear in the list on the **Manage Cloud URLs** window. Click **OK** to proceed.
6. In the **Cloud Foundry Account** window, click **Validate Account**.
7. The **Cloud Foundry Account** window is refreshed and displays a message indicating whether or not your credentials 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.



- 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 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.

```

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%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%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 on the Cloud Foundry Java Client Library, view the [source](#) on GitHub. The [domain package](#) shows the objects that can be queried and inspected.

Refer to the source of the Cloud Foundry [Maven Plugin](#) for an example of using the Cloud Foundry Java Client Library.

BOSH Custom Trusted Certificate Support

Page last updated:

Configure

Java Buildpack versions 3.7 and later support [BOSH configured custom trusted certificates](#).

Run the following command to configure support for this feature:

```
$ cf set-env APP-NAME JBP_CONFIG_CONTAINER_CERTIFICATE_TRUST_STORE '{enabled: true}'
```

Alternatively, you can modify the buildpack by setting the `enabled` property to true in `config/container_certificate_trust_store.yml`.

For more information, see the official [Java Buildpack documentation](#) for this feature.

Node.js Buildpack

Page last updated:

Use the Node.js buildpack with Node or JavaScript apps.

You need to install the [Cloud Foundry Command Line Interface \(cf CLI\)](#) 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 CLI `cf push` command push. 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` and replace `APP-NAME` with 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  
$ 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](#).

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.

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",
    "express": "3.4.8",
    "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. In order to also run your app locally, you may want to make port 3000 the default:

```
app.listen(process.env.PORT || 3000);
```

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. If you prefer to run `npm install` and create a `node_modules` folder inside of your app, this is also supported.

Solve Discovery Problems

If Cloud Foundry does not automatically detect that your app is a Node.js app, you can override the 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](#).

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](#).

Environment Variables

You can access environments variable programmatically.

For example, you can obtain `VCAP_SERVICES` like this:

```
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:/usr/bin
```

Environment Variables Defined by the Node Buildpack

Page last updated:

Elastic Runtime 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 Elastic Runtime, 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`.

.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 `[SPORT]` 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.

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:

- [Composer](#)
- [Sessions](#)
- [New Relic](#)
- [Configuration](#)
- [Deploying and Developing PHP Apps](#)
- [Tips for PHP Developers](#)

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` and supports [BOSH](#)

configured custom trusted certificates [☒](#) out of the box.

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](#) [☒](#).

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
<code>COMPOSER_VERSION</code>	The version of Composer to use. It defaults to the latest bundled with the buildpack.
<code>COMPOSER_INSTALL_OPTIONS</code>	A list of options that should be passed to <code>composer install</code> . This defaults to <code>--no-interaction --no-dev --no-progress</code> . The <code>--no-progress</code> option must be used due to the way the buildpack calls Composer.
<code>COMPOSER_VENDOR_DIR</code>	Allows you to override the default value used by the buildpack. This is passed through to Composer and instructs it where to create the <code>vendor</code> directory. Defaults to <code>{BUILD_DIR}/{LIBDIR}/vendor</code> .
<code>COMPOSER_BIN_DIR</code>	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 <code>{BUILD_DIR}/php/bin</code> .
<code>COMPOSER_CACHE_DIR</code>	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 <code>{CACHE_DIR}/composer</code> 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 the `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` this 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` this 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.

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 <code>httpd</code> , <code>nginx</code> , or <code>none</code> . This value defaults to <code>httpd</code> .
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 <code>"{PHP_70_LATEST}"</code> . See options.json .
PHP_EXTENSIONS	A list of the extensions to enable. <code>bz2</code> , <code>zlib</code> , <code>curl</code> , and <code>mcrypt</code> 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 <code>fpm</code> or <code>cli</code> . You can explicitly enable any or all of: <code>fpm</code> , <code>cli</code> , <code>cgi</code> , and <code>pear</code> .
ZEND_EXTENSIONS	A list of the Zend extensions to enable. Nothing is enabled by default.
APP_START_CMD	When the <code>WEB_SERVER</code> option is set to 'none,' this command is used to start your app. If <code>WEB_SERVER</code> and <code>APP_START_CMD</code> are not set, then the buildpack searches for <code>app.php</code> , <code>main.php</code> , <code>run.php</code> , or <code>start.php</code> (in that order). This option accepts arguments.
WEBDIR	The root directory of the files served by the web server specified in <code>WEB_SERVER</code> . Defaults to <code>htdocs</code> . Other common settings are <code>public</code> , <code>static</code> , or <code>html</code> . Path is relative to the root of your application.
LIBDIR	This path is added to PHP's <code>include_path</code> . Defaults to <code>lib</code> . 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`.

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.
```

PHP Extensions

PHP extensions are easily enabled by setting the `PHP_EXTENSIONS` or `ZEND_EXTENSIONS` option in `.bp-config/options.json`. Use these options to install bundled PHP extensions.

If an extension is already present and enabled in the compiled php, for example `intl`, you do not need to explicitly enable it through `PHP_EXTENSIONS` or `ZEND_EXTENSIONS` in `.bp-config/options.json` 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](#).

The buildpack cannot know, in advance, what kind of extension you are requesting. This is why there are two variables, and why your app will fail if you say a Zend Extension is 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 `PHP_EXTENSIONS` to `ZEND_EXTENSIONS`, 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_EXTENSIONS` to `PHP_EXTENSIONS`, 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.

Deploying and Developing PHP Apps

Page last updated:

This document is intended to guide you through the process of deploying PHP applications to Elastic Runtime. 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 Elastic Runtime. 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:

Diego release

```
$ 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].

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.

Python Buildpack

Page last updated:

Push an App

This buildpack will be automatically used if there is 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 versions of the Python runtime 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 will 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 [\[x\]](#).

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
```

`cf push` uploads your vendored dependencies. The buildpack installs them directly from the `vendor/` directory.

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> [x] for more information.

Miniconda Support (starting in buildpack version 1.5.6 [x])

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](#) [x] and [using Python 3 with miniconda](#) [x].

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](#) [x].

BOSH Configured Custom Trusted Certificate Support

Versions of Python 2.7.9 and later use certificates stored in `/etc/ssl/certs` and support [BOSH configured custom trusted certificates](#) [x] out of the box.

Help and Support

Join the `#buildpacks` channel in our [Slack community](#) [x] 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](#) [x].

You can find current information about this buildpack on the [Python buildpack release page](#) [x] in GitHub.

Ruby Buildpack

Page last updated:

Push Apps

This buildpack will be used if your app has a `Gemfile` and `Gemfile.lock` in the root directory. It will then use 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, you can use the latest version with the command:

```
cf push my_app -b https://github.com/cloudfoundry/ruby-buildpack.git
```

For more detailed information about deploying Ruby applications see the following topics:

- [Getting Started Deploying Ruby Apps](#)
- [Getting Started Deploying Ruby on Rails Apps](#)
- [Deploying a Sample Ruby on Rails App](#)
- [Configuring Rake Tasks for Deployed Apps](#)
- [Tips for Ruby Developers](#)
- [Environment Variables Defined by the Ruby Buildpack](#)
- [Configuring Service Connections for Ruby](#)

You can find the source for the buildpack on GitHub: <https://github.com/cloudfoundry/cf-buildpack-ruby>

Supported Versions

Supported Ruby versions can be found [in the release notes](#).

Specify a Ruby Version

Specific versions of the Ruby runtime can be specified in the `Gemfile`:

MRI

For MRI you can specify the version of Ruby by doing the following:

```
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 will use Ruby `2.2.5`.

For more information about the `ruby` directive for Bundler Gemfiles, see [Bundler's 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 you can specify the version of ruby by doing the following:

JRuby version `1.7.x` supports either `1.9` mode, e.g.:

```
ruby '1.9.3', :engine => 'jruby', :engine_version => '1.7.25'
```

or `2.0` mode, e.g.:

```
ruby '2.0.0', :engine => 'jruby', :engine_version => '1.7.25'
```

For Jruby version `>= 9.0`:

```
ruby '2.2.3', :engine => 'jruby', :engine_version => '9.0.5.0'
```

The buildpack only supports the stable Ruby versions, which are listed in the `manifest.yml` and [releases page](#).

If you try to use a binary that is not currently supported, staging your app will fail and you will see the following error message:

```
Could not get translated url, exited with: DEPENDENCY_MISSING_IN_MANIFEST: ...
!
! exit
!
Staging failed: Buildpack compilation step failed
```

Additionally, note that the pessimistic version operator (`<~>`) on the Gemfile `ruby` directive for JRuby is not supported by the Ruby buildpack.

Vendor App Dependencies

As stated in the [Disconnected Environments documentation](#), your application must ‘vendor’ its dependencies.

For the Ruby buildpack, use bundler:

```
cd <your app dir>
bundle package --all
```

`cf push` uploads your vendored dependencies. The buildpack will compile any dependencies requiring compilation while staging your application.

Buildpack Logging and Application Logging

The 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` and supports [BOSH configured custom trusted certificates](#) out of the box.

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 GitHub repository](#).

You can find current information about this buildpack on the Ruby buildpack [release page](#) in GitHub.

Getting Started Deploying Ruby Apps

Page last updated:

This guide is intended to walk you through deploying a Ruby app to Elastic Runtime. 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 Elastic Runtime, 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 the [Apps Manager](#) to perform this task.

Elastic Runtime supports two types of service instances:

- Managed services integrate with Elastic Runtime through service brokers that offer services and plans and manage the service calls between Elastic Runtime 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: MySQL and 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` 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`. 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, Elastic Runtime 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

Elastic Runtime uses the default standard Ruby web server library, WEBrick, for Ruby and RoR apps. However, Elastic Runtime 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 Elastic Runtime 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 Elastic Runtime 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/clouduserl/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 Elastic Runtime!

Use the cf CLI or the [Apps Manager](#) to review information and administer your app and your Elastic Runtime 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 v6](#) topic.

Note: You cannot perform certain tasks in the CLI or the [Apps Manager](#) because these are commands that only a Elastic Runtime administrator can run. If you are not a Elastic Runtime 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 Elastic Runtime. 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 the [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

Elastic Runtime requires that each app that you deploy has a unique URL. Otherwise, the new app URL collides with an existing app URL and Elastic Runtime 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 (RoR) app to Elastic Runtime. To deploy a sample RoR app, refer to the [Deploy a Sample Ruby on Rails App](#) topic.

 **Note:** Ensure that your RoR app runs locally before continuing with this procedure.

Prerequisites

- A Rails 4.x app that runs locally
- [Bundler](#) configured on your workstation
- Intermediate to advanced RoR knowledge
- The [Cloud Foundry Command Line Interface \(cf CLI\)](#)

Step 1: Create and Bind a Service Instance for a RoR Application

This section describes using the CLI to configure a PostgreSQL managed service instance for an app. 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.

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.

Bind a Service Instance

When you bind an app to a service instance, Elastic Runtime 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 whether they support this functionality.

To bind a service to an application, run `cf bind-service APPLICATION SERVICE_INSTANCE`.

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

Elastic Runtime uses the default standard Ruby web server library, WEBrick, for Ruby and RoR apps. However, Elastic Runtime 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.

Step 3: Create the Database Configuration

For 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 will be automatically set based on [bound service instances](#).

For Rails versions 4.1.0 and later, the Ruby buildpack will not modify the `config/database.yml` file.

When the app starts, the [standard Rails behavior](#) for `DATABASE_URL` and `config/database.yml` applies.

Step 4: 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 Elastic Runtime instance](#).

Step 5: Deploy Your App

From the root directory of your application, run `cf push APP-NAME --random-route` 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 Elastic Runtime 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.

To view log activity while the app deploys, launch a new terminal window and run `cf logs APP-NAME`.

In the terminal output of the `cf push` command, the `urls` field of the `App started` block contains the app URL. This is the `HOST_NAME` you specified with the `-n` flag plus the domain `shared-domain.example.com`. Once your app deploys, use this URL to access the app online.

Next Steps

You've deployed an app to Elastic Runtime! Consult the sections below for information about what to do next.

Test a Deployed App

Use the cf CLI or the [Apps Manager](#) to review information and administer your app and your Elastic Runtime 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.

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 v6](#) 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.

Deploy a Sample Ruby on Rails Application

Page last updated:

This topic guides the reader through deploying a sample Ruby on Rails app to Elastic Runtime .

Prerequisites

In order to deploy a sample Ruby on Rails app, you must have the following:

- A working PCF [deployment](#)
- [Cloud Foundry CLI](#)
- Cloud Foundry username and password with [Space Developer permissions](#). See your [Org Manager](#) if you require permissions.

Step 1: Clone the App

Run the following terminal command to create a local copy of the rails_sample_app.

```
$ git clone https://github.com/cloudfoundry-samples/rails_sample_app
```

The newly created directory contains a `manifest.yml` file, which assists CF with deploying the app. See [Deploying with Application Manifests](#) for more information.

Step 2: Log in and Target the API Endpoint

1. Run the following terminal command to log in and target the API endpoint of your deployment. For more information, see the [Identifying the API Endpoint for your Elastic Runtime Instance topic](#).

```
$ cf login -a YOUR-API-ENDPOINT
```

2. Use your credentials to log in, and to select a [Space and Org](#).

 **Note:** The API endpoint must be entered in the format `https://api.IP-ADDRESS`, where IP-ADDRESS is the IP address of your API endpoint.

Step 3: Create a Service Instance

Run the following terminal command to create a PostgreSQL service instance for the sample app. Our service instance is `rails-postgres`. It uses the `elephantsql` service and the `turtle` plan.

```
$ cf create-service elephantsql turtle rails-postgres
Creating service rails-postgres in org YOUR-ORG / space development as clouduser@example.com....
OK
```

The manifest for the rails_sample_app contains a `services` sub-block in the `applications` block. The Cloud Foundry Command Line Interface tool (cf CLI) binds the service to the 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
```

Step 4: Deploy the App

Make sure you are in the `rails_sample_app` directory. Run the following terminal command to deploy the app:

```
$ cf push rails_sample_app
```

`cf push rails_sample_app` creates a URL route to your application in the form HOST.DOMAIN. In this example, HOST is `rails_sample_app`. Administrators specify the DOMAIN. For example, for the DOMAIN `shared-domain.example.com`, running `cf push rails_sample_app` creates the URL `rails_sample_app.shared-domain.example.com`.

The example below shows the terminal output when deploying the `rails_sample_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 `rails-postgres` service and follows the information in the manifest to start one instance of the app with 256M of RAM. After the app starts, the output displays the health and status of the app.

```
$ cf push rails_sample_app
Using manifest file ~/workspace/rails_sample_app/manifest.yml

Updating app rails_sample_app in org Cloud-Apps / space development as cloudunder@example.com...
OK

Using route rails_sample_app.shared-domain.example.com
Uploading rails_sample_app...
Uploading app files from: ~/workspace/rails_sample_app
Uploading 445.7K, 217 files
OK
Binding service rails-postgres to app rails_sample_app in org Cloud-Apps / space development as cloudunder@example.com...
OK

Starting app rails_sample_app in org Cloud-Apps / space development as cloudunder@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_app in org Cloud-Apps / space development as cloudunder@example.com...
OK

requested state: started
instances: 1/1
usage: 256M x 1 instances
urls: rails_sample_app.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
```

 **Note:** If you want to view log activity while the app deploys, launch a new terminal window and run `cf logs rails_sample_app`.

Step 5: Verify the App

Verify that the app is running by browsing to the URL generated in the output of the previous step. In this example, navigating to `rails_sample_app.shared-domain.example.com` verifies that the app is running.

You've now pushed an app to Elastic Runtime! For more information about this topic, see the [Deploy an Application](#) topic.

Configure Rake Tasks for Deployed Apps

Page last updated:

For Elastic Runtime to automatically invoke a Rake task while a Ruby or Ruby on Rails app is deployed, you must:

- 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 of 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
```

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 will precompile 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.

If you have previously deployed the app, the app manifest should already exist. There are two ways to create a manifest. You can manually create the file and save it in the app root directory before you deploy the app for the first time. If you do not manually create the manifest file, the Cloud Foundry Command Line Interface (cf CLI) prompts you to supply deployment settings when you first push the app, and will create and save the manifest file for you, with the settings you specified interactively. 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 will use Sidekiq.

First, create a Rails app with an arbitrary model called “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
```

Create a worker (in app/workers) 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 will 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. The easiest 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 will create a new Sidekiq job which will be queued in Redis, then picked up by the worker app. The browser is then redirected to `/thing`, which will show 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_url` 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`

Environment Variables Defined by the Ruby Buildpack

Page last updated:

Elastic Runtime provides configuration information to applications 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 Elastic Runtime, 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
<code>BUNDLE_BIN_PATH</code>	The directory where Bundler installs binaries. Example: <code>BUNDLE_BIN_PATH:/home/vcap/app/vendor/bundle/ruby/1.9.1/gems/bundler-1.3.2/bin/bundle</code>
<code>BUNDLE_GEMFILE</code>	The path to the Gemfile for the app. Example: <code>BUNDLE_GEMFILE:/home/vcap/app/Gemfile</code>
<code>BUNDLE_WITHOUT</code>	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: <code>BUNDLE_WITHOUT=assets</code>
<code>DATABASE_URL</code>	Cloud Foundry examines the <code>database_uri</code> for bound services to see if they match known database types. If known relational database services are bound to the app, then the <code>DATABASE_URL</code> environment variable is set to the first services in the list. If your application requires that <code>DATABASE_URL</code> 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) <code>cf set-env</code> command to set this variable manually. Example: <pre>\$ cf set-env my_app_name DATABASE_URL mysql://example-database-connection-string</pre>
<code>GEM_HOME</code>	The directory where gems are installed. Example: <code>GEM_HOME:/home/vcap/app/vendor/bundle/ruby/1.9.1</code>
<code>GEM_PATH</code>	The directory where gems can be found. Example: <code>GEM_PATH=/home/vcap/app/vendor/bundle/ruby/1.9.1:</code>
<code>RACK_ENV</code>	The Rack deployment environment, which governs the middleware loaded to run the app. Valid values are <code>development</code> , <code>test</code> , and <code>production</code> . Example: <code>RAILS_ENV=production</code>
<code>RUBYOPT</code>	Defines command-line options passed to Ruby interpreter. Example: <code>RUBYOPT: -I/home/vcap/app/vendor/bundle/ruby/1.9.1/gems/bundler-1.3.2/lib -rbundler/setup</code>

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-app-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/exampledbs"  
      }  
    }  
  ]  
}
```

Based on this `VCAP_SERVICES`, Cloud Foundry creates the following `DATABASE_URL` environment variable:

```
DATABASE_URL = postgres://exampleuser:examplepass@babar.elephantsql.com:5432/exampledbs
```

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://lraa:e6B-X@p-mysql|provider.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 InitiateTerminatorT1000Deployment: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
```

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

Elastic Runtime 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 64 M` 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.

Step	Action
1	Create and move into a root directory for the sample app in your workspace: <pre>\$ mkdir sample \$ cd sample</pre>
2	Create an <code>index.html</code> file that contains some text: <pre>\$ echo 'Hello World' > index.html</pre>
3	Create an empty file named <code>Staticfile</code> : <pre>\$ touch Staticfile</pre>
4	Use the <code>cf login</code> command to log in to Elastic Runtime. For more information, see the Login section of the <i>Getting Started with the cf CLI</i> documentation. <pre>\$ cf login</pre>
5	Push the sample app:

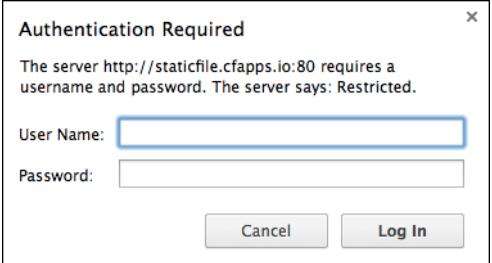
	\$ cf push hello -m 64M
6	<p>Find the URL of the app in the output. A fragment of output is shown below:</p> <pre>Creating app hello in org sample-org / space sample-space as username@example.com... OK ... requested state: started instances: 1/1 usage: 64M x 1 instances urls: hello.example.com</pre>
7	Navigate to the URL to see the sample app running.

Configuring the Buildpack and Pushing the App

This section describes various ways you can configure the Staticfile buildpack options and how to push your Staticfile app.

Configuration Options

This table describes some of the aspects of the Staticfile buildpack that you can configure.

Options	Description
Alternative root	Allows you to specify a root directory other than the default, which is <code>index.html</code> . For example, you can specify that the buildpack serves <code>index.html</code> and all other assets from an alternate folder where your HTML/CSS/JavaScript files exist, such as <code>dist/</code> or <code>public/</code> .
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 <code>index.html</code> , 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 <code>/some/path1</code> instead of <code>/some#path1</code> .
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 <code>.</code>) are not served by this buildpack.
HTTP Strict Transport Security	Causes NGINX to respond to all requests with the header <code>Strict-Transport-Security: max-age=31536000</code> . This forces browsers to make all subsequent requests over HTTPS.
Custom Location configuration	<p>To customize the <code>location</code> block of the NGINX configuration file, follow these steps:</p> <ol style="list-style-type: none"> 1. Create a file with location scoped NGINX directives. See the following example, which causes visitors of your site to receive the <code>X-MySiteName</code> HTTP header: <p>File: <code>nginx/conf/includes/custom_header.conf</code></p> <p>Content: <code>add_header X-MySiteName BestSiteEver;</code></p> 2. Set the <code>location_include</code> variable in your Staticfile to the path of the file from the previous step: <pre>... location_include: includes/*.conf ...</pre> <p>For information about NGINX directives, see NGINX documentation.</p>
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.

If you want to...	Then...	More information
serve <code>index.html</code> and other assets from a location other than the root directory	add this line to the <code>Staticfile</code> : <pre>root: YOUR-DIRECTORY-NAME</pre> For example, <code>root: public</code>	Alternative root
display a directory list instead of the standard 404 error	add this line to the <code>Staticfile</code> : <pre>directory: visible</pre>	Directory list
enable SSI	add this line to the <code>Staticfile</code> : <pre>ssi: enabled</pre>	SSI
enable pushstate routing	add this line to the <code>Staticfile</code> : <pre>pushstate: enabled</pre>	Pushstate routing
disable gzip_static and gunzip modules	add this line to the <code>Staticfile</code> : <pre>directory: visible</pre>	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 <code>Staticfile.auth</code> in the root directory or alternative root directory, and add one or more user/password lines to it. For example, <pre>bob:\$apr1\$DuUQEOp8\$ZccZCHQE1NSjrgewSFC0 alice:\$apr1\$4IRQGcd/\$UMFLnIHSD9ZHJ86TR4zx</pre>	Basic authentication
use a proxy for downloading	set the <code>http_proxy</code> and <code>https_proxy</code> environment variables.	Proxy support

dependencies during staging	set the <code>FORCE_HTTPS</code> environment variable to <code>true</code> .	
force HTTPS	<p>Note: Do not enable <code>FORCE_HTTPS</code> 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.</p>	Force HTTPS
host dot files	To enable serving hidden (dot files), use <code>host_dot_files: true</code> in the <code>Staticfile</code> .	Dot Files
force the receiving browser to make subsequent requests over HTTPS	add this line to the <code>Staticfile</code> : <pre>http_strict_transport_security: true</pre> <p>Note: Because this setting persists in browsers for a long time, only enable this setting after ensuring you have completed your configuration.</p>	HTTP Strict Transport Security
make additional configuration changes to NGINX	add <code>nginx.conf</code> and <code>mime.types</code> 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.	<p>Use the <code>cf push APP_NAME -m 64M</code> command to push your app. Replace <code>APP_NAME</code> with the name you want to give your application. For example:</p> <pre>\$ 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</pre> <p>Or, if you do not have the buildpack, or the installed version is out-of-date, use the <code>-b</code> option to specify the buildpack as follows:</p> <pre>cf push APP_NAME -b https://github.com/cloudfoundry/staticfile-buildpack.git</pre>
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](#).

Using 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:

- [Buildpack Detection](#)
- [Proxy Usage](#)
- [Supported Binary Dependencies](#)
- [Configuring a Production Server](#)

Buildpack Detection

Page last updated:

When you push an app, Cloud Foundry uses a detection process to determine which 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 (identified 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 `cf push` fails with the following error:

```
None of the buildpacks detected a compatible application
Exit status 222
Staging failed: Exited with status 222

FAILED
NoAppDetectedError
```

Proxy Usage

Page last updated:

Buildpacks can use proxies via the `http_proxy` and/or `https_proxy` environment variables. These should be set to the proxy hostname and/or port.

These can be set via the [manifest.yml](#) file or `cf set-env`.

All of the buildpacks will automatically utilize these proxy environment variables correctly. If any of them contact the internet during staging, it will be through the proxy host. The binary buildpack will not use a proxy because it does not use the internet at all during staging.

```
--  
env:  
http_proxy: http://proxy-site.com:3000  
https_proxy: https://proxy-site.com:3003
```

Note: Whilst many applications will use the `http_proxy` and `https_proxy` environment variables at runtime, this is entirely dependent on your application, the buildpack does not add any 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
```

Configuring a Production Server

Page last updated:

This topic describes how to configure a production server for your apps.

When you deploy an app, Elastic Runtime determines the command used to start the app through the following process:

1. If the developer uses the command `cf push -c COMMAND`, then Elastic Runtime uses `COMMAND` to start the app.
2. If the developer creates a file called a Procfile, Elastic Runtime 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 Elastic Runtime 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 to 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

Elastic Runtime uses the default standard Ruby web server library WEBrick for Ruby and Ruby on Rails apps. However, Elastic Runtime can support a more robust production web server, such as Phusion Passenger, Puma, Thin, or Unicorn.

To instruct Elastic Runtime 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
```

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.

About 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:
 - [Custom Buildpacks](#)
 - [Packaging Dependencies for Offline Buildpacks](#)

About 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.

Custom Buildpacks

Page last updated:

Buildpack Scripts

A buildpack repository contains three main scripts, situated in a folder named `bin`.

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 helpful information to `STDOUT`.

The following is an example `detect` script written in Ruby 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 decided by the buildpack developer. These additional details include buildpack versioning information and a detailed 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-
```

bin/compile

The `compile` script builds a droplet by packaging the app dependencies, assuring that the app has all the necessary components needed to run.

The script is run with two arguments: the `build` directory for the app and 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 through the cf CLI to the user.

The following is an example of a simple `compile` script:

```
#!/usr/bin/env ruby
#sync output
$stdout.sync = true
build_path = ARGV[0]
cache_path = ARGV[1]
install_ruby
private
def install_ruby
  puts "Installing Ruby"
  # !!! build tasks go here !!!
  # download ruby
  # install ruby
end
```

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 `web` type of apps are 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/
logs/
tmp/
staging_info.yml
```

The `app` directory contains `BUILD_DIR` contents, and `staging_info.yml` contains the staging metadata saved in the droplet.

Package Custom Buildpacks

Cloud Foundry buildpacks work with limited or no Internet connectivity. A Cloud Foundry operator can use the [buildpack packager](#) RubyGem

to give 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 will download and add 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, you can 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 repo](#).

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.

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 Cloud Foundry [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: 1.3.1
```

- 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: 1.6.3` for the `go` dependency, but lists `version: 1.5.4` under `dependencies`.

```
# Incorrect; will fail to package
default_versions:
- name: go
  version: 1.6.3

dependencies:
- name: go
  version: 1.5.4
uri: https://storage.googleapis.com/golang/go1.5.4.linux-amd64.tar.gz
md5: 27b1c469797292064c65c995ff30386
cf_stacks:
- cflinuxfs2
```

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, Cloud Foundry 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 or shas in a git repository, as follows:

```
$ cf push my-new-app -b https://github.com/johndoe/my-buildpack.git#v1.4.2
```

```
$ cf push my-new-app -b https://github.com/johndoe/my-buildpack.git#a2951e298bd22732fceb3968d541015120dbaf93
```

The app will then be deployed to Cloud Foundry, 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 custom offline buildpacks.

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

$ # 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

$ # 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.

- 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
```

- Use the `git add` command to stage your changes:

```
$ git add .
```

- 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]"
```

- Run `git push` to push your changes to the remote origin.

```
$ git push
```

- 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.

- 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
```

- Edit the `manifest.yml` file for the buildpack to add or remove dependencies.

```
dependencies:  
- name: ruby  
  version: 2.3.0  
  md5: 535342030a11abeb11497824bf642bf2  
  uri: https://pivotals-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.

- 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
```

- 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.

- 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
```

- 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](#) [link](#)

Godep:

- Built from:** A source code `.tar.gz` file from the [Godep Github releases](#) [page](#)
- Verified with:** The SHA256 of the source **Example:** [Automated enqueueing of binary build for Godep 72](#) [link](#)

 **Note:** The `buildpacks-ci` [link](#) `binary-builder` [link](#) 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 enqueueing 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:** [Enqueueing binary build for Nginx 1.11.0](#)

Binary Buildpack

The Binary buildpack has no dependencies.

CF Buildpack Team CI

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 HEAD:{master,develop}
```

8. Merge your changes to the [master](#) branch:

```
$ git checkout master  
$ git merge 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 [detect-new-buildpack-and-upload-artifacts](#) job in the pipeline for the updated buildpack. This job creates a cached and uncached buildpack and uploads them to an AWS S3 bucket.
2. 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 \(BRATS\)](#) [🔗](#).

3. The `buildpack-release` job triggers and creates a tag for the new version.
4. If you are using [Pivotal Tracker](#), paste the links for the `specs-edge-master`, `specs-lts-master`, and `buildpack-release` 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 [machete](#) and [buildpack-packager](#), 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

- [Overview](#)
- [Service Broker API](#)
- [Service Broker API Release Notes](#)
- [Managing Service Brokers](#)
- [Access Control](#)
- [Catalog Metadata](#)
- [Dashboard Single Sign-On](#)
- [Example Service Brokers](#)
- [Binding Credentials](#)
- [Application Log Streaming](#)
- [Route Services](#)
- [Supporting Multiple Cloud Foundry Instances](#)

Overview

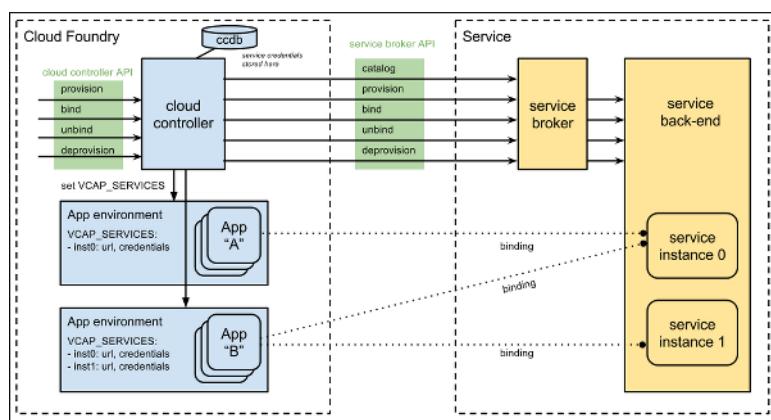
Page last updated:

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

Service Broker API v2.11

Table of Contents

- [API Release Notes](#)
- [Changes](#)
 - [Change Policy](#)
 - [Changes Since v2.10](#)
- [API Overview](#)
- [API Version Header](#)
- [Authentication](#)
- [Catalog Management](#)
 - [Adding a Broker to the Platform](#)
- [Synchronous and Asynchronous Operations](#)
 - [Synchronous Operations](#)
 - [Asynchronous Operations](#)
- [Polling Last Operation](#)
 - [Polling Interval and Duration](#)
- [Provisioning](#)
- [Updating a Service Instance](#)
- [Binding](#)
 - [Types of Binding](#)
- [Unbinding](#)
- [Deprovisioning](#)
- [Broker Errors](#)
- [Orphans](#)

Changes

Change Policy

- Existing endpoints and fields will not be removed or renamed.
- New optional endpoints, or new HTTP methods for existing endpoints, may be added to enable support for new features.
- New fields may be added to existing request/response messages. These fields must be optional and should be ignored by clients and servers that do not understand them.

Changes Since v2.10

- Add `bindable` field to [Plan Object](#) to allow services to have both bindable and non-bindable plans.

API Overview

The Service Broker API defines an HTTP interface between the services marketplace of a platform and service brokers.

The service broker is the component of the service that implements the Service Broker API, for which a platform's marketplace is a client. Service brokers are responsible for advertising a catalog of service offerings and service plans to the marketplace, and acting on requests from the marketplace for provisioning, binding, unbinding, and deprovisioning.

In general, provisioning reserves a resource on a service; we call this reserved resource a service instance. What a service instance represents can vary by service. Examples include a single database on a multi-tenant server, a dedicated cluster, or an account on a web application.

What a binding represents may also vary by service. In general creation of a binding either generates credentials necessary for accessing the resource or provides the service instance with information for a configuration change.

A platform marketplace may expose services from one or many service brokers, and an individual service broker may support one or many platform marketplaces using different URL prefixes and credentials.

API Version Header

Requests from the platform to the service broker must contain a header that declares the version number of the Service Broker API that the marketplace will use:

X-Broker-Api-Version: 2.11

The version numbers are in the format `MAJOR.MINOR`, using semantic versioning such that 2.10 comes before 2.11.

This header allows brokers to reject requests from marketplaces for versions they do not support. While minor API revisions will always be additive, it is possible that brokers depend on a feature from a newer version of the API that is supported by the platform. In this scenario the broker may reject the request with `412 Precondition Failed` and provide a message that informs the operator of the required API version.

Authentication

The marketplace must authenticate with the service broker using HTTP basic authentication (the `Authorization:` header) on every request. The broker is responsible for validating the username and password and returning a `401 Unauthorized` message if credentials are invalid. It is recommended that brokers support secure communication from platform marketplaces over TLS.

Catalog Management

The first endpoint that a broker must implement is the service catalog.

The platform marketplace fetches this endpoint from all brokers in order to present an aggregated user-facing catalog.

Warnings for broker authors:

- Be cautious removing services and plans from their catalogs, as platform marketplaces may have provisioned service instances of these plans. Consider your deprecation strategy.
- Do not change the ids of services and plans. This action is likely to be evaluated by a platform marketplace as a removal of one plan and addition of another. See above warning about removal of plans.

The following sections describe catalog requests and responses in the Service Broker API.

Request

Route

GET /v2/catalog

cURL

```
$ curl -H "X-Broker-API-Version: 2.11" http://username:password@broker-url/v2/catalog
```

Response

Status Code	Description
200 OK	The expected response body is below.

Body - Schema of Service Objects

CLI and web clients have different needs with regard to service and plan names. A CLI-friendly string is all lowercase, with no spaces. Keep it short – imagine a user having to type it as an argument for a longer command. A web-friendly display name is camel-cased with spaces and punctuation supported.

Response field	Type	Description
services*	array-of-service-objects	Schema of service objects defined below.

Service Objects

Response field	Type	Description
name*	string	A CLI-friendly name of the service. All lowercase, no spaces. This must be globally unique within a platform marketplace.
id*	string	An identifier used to correlate this service in future requests to the broker. This must be globally unique within a platform marketplace. Using a GUID is recommended.
description*	string	A short description of the service.
tags	array-of-strings	Tags provide a flexible mechanism to expose a classification, attribute, or base technology of a service, enabling equivalent services to be swapped out without changes to dependent logic in applications, buildpacks, or other services. E.g. mysql, relational, redis, key-value, caching, messaging, amqp.
requires	array-of-strings	A list of permissions that the user would have to give the service, if they provision it. The only permissions currently supported are <code>syslog_drain</code> , <code>route_forwarding</code> and <code>volume_mount</code> .
bindable*	boolean	Specifies whether instances of the service can be bound to applications. This specifies the default for all plans of this service. Plans can override this field (see Plan Object).
metadata	object	A list of metadata for a service offering.
dashboard_client	object	Contains the data necessary to activate the Dashboard SSO feature for this service
plan_updateable	boolean	Whether the service supports upgrade/downgrade for some plans. Please note that the misspelling of the attribute <code>plan_updatable</code> to <code>plan_updateable</code> was done by mistake. We have opted to keep that misspelling instead of fixing it and thus breaking backward compatibility.
plans*	array-of-objects	A list of plans for this service, schema is defined below.

Dashboard Client Object

Response field	Type	Description
id	string	The id of the Oauth client that the dashboard will use.
secret	string	A secret for the dashboard client
redirect_uri	string	A URI for the service dashboard. Validated by the OAuth token server when the dashboard requests a token.

Plan Object

Response field	Type	Description
id*	string	An identifier used to correlate this plan in future requests to the broker. This must be globally unique within a platform marketplace. Using a GUID is recommended.
name*	string	The CLI-friendly name of the plan. Must be unique within the service. All lowercase, no spaces.

Response field*	Type	Description
metadata	object	A list of metadata for a service plan.
free	boolean	When false, instances of this plan have a cost. The default is true
bindable	boolean	Specifies whether instances of the service plan can be bound to applications. This field is optional. If specified, this takes precedence over the bindable attribute of the service. If not specified, the default is derived from the service.

* Fields with an asterisk are required.

```
{
  "services": [
    {
      "name": "fake-service",
      "id": "acb56d7c-XXXX-XXXX-XXXX-feb140a59a66",
      "description": "fake service",
      "tags": ["no-sql", "relational"],
      "requires": ["route_forwarding"],
      "max_db_per_node": 5,
      "bindable": true,
      "metadata": {
        "provider": {
          "name": "The name"
        },
        "listing": {
          "imageUrl": "http://example.com/cat.gif",
          "blurb": "Add a blurb here",
          "longDescription": "A long time ago, in a galaxy far far away..."
        },
        "displayName": "The Fake Broker"
      },
      "dashboard_client": {
        "id": "398e2f8e-XXXX-XXXX-XXXX-19a71ecbcf64",
        "secret": "277cab0-XXXX-XXXX-XXXX-7822c0a90e5d",
        "redirect_uri": "http://localhost:1234"
      },
      "plan_updateable": true,
      "plans": [
        {
          "name": "fake-plan-1",
          "id": "d3031751-XXXX-XXXX-XXXX-a42377d3320e",
          "description": "Shared fake Server, 5tb persistent disk, 40 max concurrent connections",
          "max_storage_tb": 5,
          "metadata": {
            "costs": [
              {
                "amount": {
                  "usd": 99.0
                },
                "unit": "MONTHLY"
              },
              {
                "amount": {
                  "usd": 0.99
                },
                "unit": "1GB of messages over 20GB"
              }
            ],
            "bullets": [
              {
                "content": "Shared fake server"
              },
              {
                "content": "5 TB storage"
              },
              {
                "content": "40 concurrent connections"
              }
            ]
          },
          {
            "name": "fake-plan-2",
            "id": "0f4008b5-XXXX-XXXX-XXXX-dace631cd648",
            "description": "Shared fake Server, 5tb persistent disk, 40 max concurrent connections. 100 async",
            "max_storage_tb": 5,
            "metadata": {
              "costs": [
                {
                  "amount": {
                    "usd": 199.0
                  },
                  "unit": "MONTHLY"
                },
                {
                  "amount": {
                    "usd": 0.99
                  },
                  "unit": "1GB of messages over 20GB"
                }
              ]
            }
          }
        }
      ]
    }
  ]
}
```

```
        }
    ],
    "bullets": [
        {
            "content": "40 concurrent connections"
        }
    ]
}
}
```

Adding a Broker to the Platform

After implementing the first endpoint `GET /v2/catalog` documented [above](#), you must register the service broker with your platform to make your services and plans available to end users.

Synchronous and Asynchronous Operations

Broker clients expect prompt responses to all API requests in order to provide users with fast feedback. Service broker authors should implement their brokers to respond promptly to all requests but must decide whether to implement synchronous or asynchronous responses. Brokers that can guarantee completion of the requested operation with the response should return the synchronous response. Brokers that cannot guarantee completion of the operation with the response should implement the asynchronous response.

Providing a synchronous response for a provision, update, or bind operation before actual completion causes confusion for users as their service may not be usable and they have no way to find out when it will be. Asynchronous responses set expectations for users that an operation is in progress and can also provide updates on the status of the operation.

Support for synchronous or asynchronous responses may vary by service offering, even by service plan.

Synchronous Operations

To execute a request synchronously, the broker need only return the usual status codes: `201 CREATED` for provision and bind, and `200 OK` for update, unbind, and deprovision.

Brokers that support synchronous responses for provision, update, and delete can ignore the `accepts_incomplete=true` query parameter if it is provided by the client.

Asynchronous Operations

 **Note:** Asynchronous operations are currently supported only for provision, update, and deprovision.

For a broker to return an asynchronous response, the query parameter `accepts_incomplete=true` must be included the request. If the parameter is not included or is set to `false`, and the broker cannot fulfill the request synchronously (guaranteeing that the operation is complete on response), then the broker should reject the request with the status code `422 UNPROCESSABLE ENTITY` and the following body:

```
{
    "error": "AsyncRequired",
    "description": "This service plan requires client support for asynchronous service operations."
}
```

If the query parameter described above is present, and the broker executes the request asynchronously, the broker must return the asynchronous response `202 ACCEPTED`. The response body should be the same as if the broker were serving the request synchronously.

An asynchronous response triggers the platform marketplace to poll the endpoint `GET /v2/service_instances/:guid/last_operation` until the broker indicates that the requested operation has succeeded or failed. Brokers may include a status message with each response for the `last_operation` endpoint that provides visibility to end users as to the progress of the operation.

Blocking Operations

The marketplace must ensure that service brokers do not receive requests for an instance while an asynchronous operation is in progress. For example, if a broker is in the process of provisioning an instance asynchronously, the marketplace must not allow any update, bind, unbind, or deprovision requests to be made through the platform. A user who attempts to perform one of these actions while an operation is already in progress must receive an HTTP 400 response with the error message: `Another operation for this service instance is in progress.`

Polling Last Operation

When a broker returns status code `202 ACCEPTED` for `provision`, `update`, or `deprovision`, the platform will begin polling the `/v2/service_instances/:guid/last_operation` endpoint to obtain the state of the last requested operation. The broker response must contain the field `state` and an optional field `description`.

Valid values for `state` are `in progress`, `succeeded`, and `failed`. The platform will poll the `last_operation` endpoint as long as the broker returns `"state": "in progress"`. Returning `"state": "succeeded"` or `"state": "failed"` will cause the platform to cease polling. The value provided for `description` will be passed through to the platform API client and can be used to provide additional detail for users about the progress of the operation.

Request

Route

```
GET /v2/service_instances/:instance_id/last_operation
```

Parameters

The request provides these query string parameters as useful hints for brokers.

Query-String Field	Type	Description
<code>service_id</code>	string	ID of the service from the catalog.
<code>plan_id</code>	string	ID of the plan from the catalog.
<code>operation</code>	string	A broker-provided identifier for the operation. When a value for <code>operation</code> is included with asynchronous responses for <code>Provision</code> , <code>Update</code> , and <code>Deprovision</code> requests, the broker client should provide the same value using this query parameter as a URL-encoded string.

 **Note:** Although the request query parameters `service_id` and `plan_id` are not required, the platform should include them on all `last_operation` requests it makes to service brokers.

cURL

```
$ curl http://username:password@broker-url/v2/service_instances/:instance_id/last_operation
```

Response

Status Code	Description
<code>200 OK</code>	The expected response body is below.
<code>410 GONE</code>	Appropriate only for asynchronous delete operations. The platform should consider this response a success and remove the resource from its database. The expected response body is <code>{}</code> . Returning this while the platform is polling for create or update operations should be interpreted as an invalid response and the platform should continue polling.

Responses with any other status code should be interpreted as an error or invalid response. The platform should continue polling until the broker

returns a valid response or the [maximum polling duration](#) is reached. Brokers may use the `description` field to expose user-facing error messages about the operation state; for more info see [Broker Errors](#).

Body

All response bodies must be a valid JSON Object ([8](#)). This is for future compatibility; it will be easier to add fields in the future if JSON is expected rather than to support the cases when a JSON body may or may not be returned.

For success responses, the following fields are valid.

Response field	Type	Description
state*	string	Valid values are <code>in progress</code> , <code>succeeded</code> , and <code>failed</code> . While <code>"state": "in progress"</code> , the platform should continue polling. A response with <code>"state": "succeeded"</code> or <code>"state": "failed"</code> should cause the platform to cease polling.
description	string	Optional field. A user-facing message displayed to the platform API client. Can be used to tell the user details about the status of the operation.

* Fields with an asterisk are required.

```
{
  "state": "in progress",
  "description": "Creating service (10% complete)."
}
```

Polling Interval and Duration

The frequency and maximum duration of polling may vary by platform client. If a platform has a max polling duration and this limit is reached, the platform will cease polling and the operation state will be considered `failed`.

Provisioning

When the broker receives a provision request from the platform, it should take whatever action is necessary to create a new resource. What provisioning represents varies by service and plan, although there are several common use cases. For a MySQL service, provisioning could result in an empty dedicated database server running on its own VM or an empty schema on a shared database server. For non-data services, provisioning could just mean an account on a multi-tenant SaaS application.

Request

Route

```
PUT /v2/service_instances/:instance_id
```

The `:instance_id` of a service instance is provided by the platform. This ID will be used for future requests (bind and deprovision), so the broker must use it to correlate the resource it creates.

Body

Request field	Type	Description
service_id*	string	The ID of the service (from the catalog). Must be globally unique.
plan_id*	string	The ID of the plan (from the catalog) for which the service instance has been requested. Must be unique to a service.
parameters	JSON object	Configuration options for the service instance.

Request field	Type	Description
accepts_incomplete	boolean	A value of true indicates that the marketplace and its clients support asynchronous broker operations. If this parameter is not included in the request, and the broker can only provision an instance of the requested plan asynchronously, the broker should reject the request with a 422 as described below.
organization_guid*	string	The platform GUID for the organization under which the service is to be provisioned. Although most brokers will not use this field, it may be helpful for executing operations on a user's behalf.
space_guid*	string	The identifier for the project space within the platform organization. Although most brokers will not use this field, it may be helpful for executing operations on a user's behalf.

* Fields with an asterisk are required.

```
{
  "service_id": "service-guid-here",
  "plan_id": "plan-guid-here",
  "organization_guid": "org-guid-here",
  "space_guid": "space-guid-here",
  "parameters": {
    "parameter1": 1,
    "parameter2": "foo"
  }
}
```

cURL

```
$ curl http://username:password@broker-url/v2/service_instances/:instance_id -d '{
  "service_id": "service-guid-here",
  "plan_id": "plan-guid-here",
  "organization_guid": "org-guid-here",
  "space_guid": "space-guid-here",
  "parameters": {
    "parameter1": 1,
    "parameter2": "foo"
  }
}' -X PUT -H "X-Broker-API-Version: 2.11" -H "Content-Type: application/json"
```

Response

Status Code	Description
201 Created	Service instance has been provisioned. The expected response body is below.
200 OK	May be returned if the service instance already exists and the requested parameters are identical to the existing service instance. The expected response body is below.
202 Accepted	Service instance provisioning is in progress. This triggers the platform marketplace to poll the Service Instance Last Operation Endpoint for operation status.
409 Conflict	Should be returned if a service instance with the same id already exists but with different attributes. The expected response body is <code>{}</code> .
422 Unprocessable Entity	Should be returned if the broker only supports asynchronous provisioning for the requested plan and the request did not include <code>?accepts_incomplete=true</code> . The expected response body is: <code>{ "error": "AsyncRequired", "description": "This service plan requires client support for asynchronous service operations." }</code> , as described below.

Responses with any other status code will be interpreted as a failure. Brokers can include a user-facing message in the `description` field; for details see [Broker Errors](#).

Body

All response bodies must be a valid JSON Object (`{}`). This is for future compatibility; it will be easier to add fields in the future if JSON is expected rather than to support the cases when a JSON body may or may not be returned.

For success responses, a broker may return the following fields. For error responses, see [Broker Errors](#).

Response field	Type	Description
dashboard_url	string	The URL of a web-based management user interface for the service instance; we refer to this as a service dashboard. The URL should contain enough information for the dashboard to identify the resource being accessed (9189kdfsk0vfnku in the example below).
operation	string	For asynchronous responses, service brokers may return an identifier representing the operation. The value of this field should be provided by the broker client with requests to the Last Operation endpoint in a URL encoded query parameter.

* Fields with an asterisk are required.

```
{
  "dashboard\ url": "http://example-dashboard.example.com/9189kdfsk0vfnku",
  "operation": "task\_\_10"
}
```

Updating a Service Instance

By implementing this endpoint, service broker authors can enable users to modify two attributes of an existing service instance: the service plan and parameters. By changing the service plan, users can upgrade or downgrade their service instance to other plans. By modifying properties, users can change configuration options that are specific to a service or plan.

To enable support for the update of the plan, a broker should declare support per service by including `plan_updateable: true` in its [catalog endpoint](#).

Not all permutations of plan changes are expected to be supported. For example, a service may support upgrading from plan “shared small” to “shared large” but not to plan “dedicated”. It is up to the broker to validate whether a particular permutation of plan change is supported. If a particular plan change is not supported, the broker should return a meaningful error message in response.

Request

Route

`PATCH /v2/service_instances/:instance_id`

`:instance_id` is the global unique ID of a previously-provisioned service instance.

Body

Request Field	Type	Description
service_id*	string	The ID of the service (from the catalog). Must be globally unique.
plan_id	string	The ID of the plan (from the catalog) for which the service instance has been requested. Must be unique to a service.
parameters	JSON object	Configuration options for the service instance.
accepts_incomplete	boolean	A value of true indicates that the marketplace and its clients support asynchronous broker operations. If this parameter is not included in the request, and the broker can only provision an instance of the requested plan asynchronously, the broker should reject the request with a 422 as described below.
previous_values	object	Information about the instance prior to the update.
previous_values.service_id	string	ID of the service for the instance.
previous_values.plan_id	string	ID of the plan prior to the update.
previous_values.organization_id	string	ID of the organization specified for the instance.
previous_values.space_id	string	ID of the space specified for the instance.

* Fields with an asterisk are required.

```
{
  "service_id": "service-guid-here",
  "plan_id": "plan-guid-here",
  "parameters": {
    "parameter1": 1,
    "parameter2": "foo"
  },
  "previous_values": {
    "plan_id": "old-plan-guid-here",
    "service_id": "service-guid-here",
    "organization_id": "org-guid-here",
    "space_id": "space-guid-here"
  }
}
```

cURL

```
$ curl http://username:password@broker-url/v2/service_instances/:instance_id -d'{
  "service_id": "service-guid-here",
  "plan_id": "plan-guid-here",
  "parameters": {
    "parameter1": 1,
    "parameter2": "foo"
  },
  "previous_values": {
    "plan_id": "old-plan-guid-here",
    "service_id": "service-guid-here",
    "organization_id": "org-guid-here",
    "space_id": "space-guid-here"
  }
}' -X PATCH -H "X-Broker-API-Version: 2.11" -H "Content-Type: application/json"
```

Response

Status Code	Description
200 OK	The requests changes have been applied. The expected response body is <code>{}</code> .
202 Accepted	Service instance update is in progress. This triggers the platform marketplace to poll the Service Instance Last Operation Endpoint for operation status.
422 Unprocessable entity	May be returned if the requested change is not supported or if the request cannot currently be fulfilled due to the state of the instance (e.g. instance utilization is over the quota of the requested plan). Broker should include a user-facing message in the body; for details see Broker Errors . Additionally, a 422 can also be returned if the broker only supports asynchronous update for the requested plan and the request did not include <code>?accepts_incomplete=true</code> ; in this case the expected response body is: <code>{ "error": "AsyncRequired", "description": "This service plan requires client support for asynchronous service operations." }</code>

Responses with any other status code will be interpreted as a failure. Brokers can include a user-facing message in the `description` field; for details see [Broker Errors](#).

Body

All response bodies must be a valid JSON Object (`{}`). This is for future compatibility; it will be easier to add fields in the future if JSON is expected rather than to support the cases when a JSON body may or may not be returned.

For success responses, a broker may return the following field. Others will be ignored. For error responses, see [Broker Errors](#).

Response field	Type	Description
operation	string	For asynchronous responses, service brokers may return an identifier representing the operation. The value of this field should be provided by the broker client with requests to the Last Operation endpoint in a URL encoded query parameter.

* Fields with an asterisk are required.

```
{
  "operation": "task_10"
}
```

Binding

If `bindable:true` is declared for a service or plan in the [Catalog](#) endpoint, broker clients may request generation of a service binding.

 **Note:** Not all services must be bindable — some deliver value just from being provisioned. Brokers that offer services that are bindable should declare them as such using `bindable: true` in the [Catalog](#). Brokers that do not offer any bindable services do not need to implement the endpoint for bind requests.

Types of Binding

Credentials

Credentials are a set of information used by an application or a user to utilize the service instance. If the broker supports generation of credentials it should return `credentials` in the response for a request to create a service binding. Credentials should be unique whenever possible, so access can be revoked for each binding without affecting consumers of other bindings for the service instance.

Log Drain

There are a class of service offerings that provide aggregation, indexing, and analysis of log data. To utilize these services an application that generates logs needs information for the location to which it should stream logs. If a broker represents one of these services, it may optionally return a `syslog_drain_url` in the response for a request to create a service binding, to which logs may be streamed.

The `requires` field in the [Catalog](#) endpoint enables a platform marketplace to validate a response for create binding that includes a `syslog_drain_url`. Platform marketplaces should consider a broker's response invalid if it includes a `syslog_drain_url` and `"requires":["syslog_drain"]` is not present in the [Catalog](#) endpoint.

Route Services

There are a class of service offerings that intermediate requests to applications, performing functions such as rate limiting or authorization. To configure a service instance with behavior specific to an application's routable address, a broker client may send the address along with the request to create a binding using `"bind_resource":{"route":"some-address.com"}`.

Some platforms may support proxying of application requests to service instances. In this case the platform needs to know where to send application requests; to facilitate this, the broker may return a `route_service_url` in the response for a request to create a binding. Not all services of this type expect to receive requests proxied by the platform; some services will have been configured out-of-band to intermediate requests to applications. In this case, the broker will not return `route_service_url` in response to the create binding request. By sending `bind-resource` as described above, the platform enables dynamic configuration of a service instance already in the application request path for the route, requiring no change in the platform routing tier.

The `requires` field in the [Catalog](#) endpoint enables a platform marketplace to validate requests to create bindings. A platform may opt to reject requests to create bindings when a broker has declared `"requires":["route_forwarding"]` for a service in the catalog endpoint.

Volume Services

There are a class of services that provide network storage to applications via volume mounts in the application container. A service broker may return data required for this configuration with `volume_mount` in response to the request to create a binding.

The `requires` field in the [Catalog](#) endpoint enables a platform marketplace to validate a response for create binding that includes a `volume_mounts`. Platform marketplaces should consider a broker's response invalid if it includes a `volume_mounts` and `"requires":["volume_mount"]` is not present in the [Catalog](#) endpoint.

Request

Route

```
PUT /v2/service_instances/:instance_id/service_bindings/:binding_id
```

The `:instance_id` is the ID of a previously-provisioned service instance. The `:binding_id` is also provided by the platform. This ID will be used for future unbind requests, so the broker must use it to correlate the resource it creates.

Body

Request Field	Type	Description
service_id*	string	ID of the service from the catalog.
plan_id*	string	ID of the plan from the catalog.
app_guid	string	Deprecated in favor of <code>bind_resource.app_guid</code> . GUID of an application associated with the binding to be created.
bind_resource	JSON object	A JSON object that contains data for platform resources associated with the binding to be created. Current valid values include <code>app_guid</code> for credentials and <code>route</code> for route services .
parameters	JSON object	Configuration options for the service binding.

* Fields with an asterisk are required.

```
{
  "service_id": "service-guid-here",
  "plan_id": "plan-guid-here",
  "bind_resource": {
    "app_guid": "app-guid-here"
  },
  "parameters": {
    "parameter1-name-here": 1,
    "parameter2-name-here": "parameter2-value-here"
  }
}
```

cURL

```
$ curl http://username:password@broker-url/v2/service_instances/:instance_id/service_bindings/:binding_id -d'{
  "service_id": "service-guid-here",
  "plan_id": "plan-guid-here",
  "bind_resource": {
    "app_guid": "app-guid-here"
  },
  "parameters": {
    "parameter1-name-here": 1,
    "parameter2-name-here": "parameter2-value-here"
  }
}' -X PUT
```

Response

Status Code	Description
201 Created	Binding has been created. The expected response body is below.
200 OK	May be returned if the binding already exists and the requested parameters are identical to the existing binding. The expected response body is below.
409 Conflict	Should be returned if the requested binding already exists. The expected response body is <code>{}</code> , though the description field can be used to return a user-facing error message, as described in Broker Errors .
422 Unprocessable	Should be returned if the broker requires that <code>app_guid</code> be included in the request body. The expected response body is:

Status Code	Description
	"RequiresApp", "description": "This service supports generation of credentials through binding an application only." }

Responses with any other status code will be interpreted as a failure and an unbind request will be sent to the broker to prevent an orphan being created on the broker. Brokers can include a user-facing message in the `:description` field; for details see [Broker Errors](#).

Body

All response bodies must be a valid JSON Object (`{ }`). This is for future compatibility; it will be easier to add fields in the future if JSON is expected rather than to support the cases when a JSON body may or may not be returned.

For success responses, the following fields are supported. Others will be ignored. For error responses, see [Broker Errors](#).

Response Field	Type	Description
credentials	object	A free-form hash of credentials that may be used by applications or users to access the service.
syslog_drain_url	string	A URL to which logs may be streamed. <code>"requires": ["syslog_drain"]</code> must be declared in the Catalog endpoint or the platform should consider the response invalid.
route_service_url	string	A URL to which the platform may proxy requests for the address sent with <code>bind_resource.route</code> in the request body. <code>"requires": ["route_forwarding"]</code> must be declared in the Catalog endpoint or the platform should consider the response invalid.
volume_mounts	array-of-objects	An array of configuration for mounting volumes. <code>"requires": ["volume_mount"]</code> must be declared in the Catalog endpoint or the platform should consider the response invalid.

```
{
  "credentials": {
    "uri": "mysql://mysqluser:pass@mysqlhost:3306/dbname",
    "username": "mysqluser",
    "password": "pass",
    "host": "mysqlhost",
    "port": 3306,
    "database": "dbname"
  }
}
```

Unbinding

 **Note:** Brokers that do not provide any bindable services or plans do not need to implement this endpoint.

When a broker receives an unbind request from the marketplace, it should delete any resources associated with the binding. In the case where credentials were generated, this may result in requests to the service instance failing to authenticate.

Request

Route

```
DELETE /v2/service_instances/:instance_id/service_bindings/:binding_id
```

The `:instance_id` is the ID of a previously-provisioned service instance. The `:binding_id` is the ID of a previously provisioned binding for that instance.

Parameters

The request provides these query string parameters as useful hints for brokers.

Query-String Field	Type	Description
service_id*	string	ID of the service from the catalog.

Query String Field	Type	Description
--------------------	------	-------------

* Query parameters with an asterisk are required.

cURL

```
$ curl 'http://username:password@broker-url/v2/service_instances/:instance_id/service_bindings/:binding_id?service_id=service-id-here&plan_id=plan-id-here' -X DELETE -H "X-Broker-API-Version: 2.11"
```

Response

Status Code	Description
200 OK	Binding was deleted. The expected response body is <code>{}</code> .
410 Gone	Should be returned if the binding does not exist. The expected response body is <code>{}</code> .

Responses with any other status code will be interpreted as a failure and the binding will remain in the marketplace database. Brokers can include a user-facing message in the `description` field; for details see [Broker Errors](#).

Body

All response bodies must be a valid JSON Object (`{}`). This is for future compatibility; it will be easier to add fields in the future if JSON is expected rather than to support the cases when a JSON body may or may not be returned.

For a success response, the expected response body is `{}`.

Deprovisioning

When a broker receives a deprovision request from the marketplace, it should delete any resources it created during the provision. Usually this means that all resources are immediately reclaimed for future provisions.

Request

Route

`DELETE /v2/service_instances/:instance_id`

`:instance_id` is the identifier of a previously provisioned instance.

Parameters

The request provides these query string parameters as useful hints for brokers.

Query-String Field	Type	Description
service_id*	string	ID of the service from the catalog.
plan_id*	string	ID of the plan from the catalog.
accepts_incomplete	boolean	A value of true indicates that both the marketplace and the requesting client support asynchronous deprovisioning. If this parameter is not included in the request, and the broker can only deprovision an instance of the requested plan asynchronously, the broker should reject the request with a 422 as described below.

* Query parameters with an asterisk are required.

cURL

```
$ curl 'http://username:password@broker-url/v2/service_instances/:instance_id?service_id=service-id-here&plan_id=plan-id-here' -X DELETE -H "X-Broker-API-Version: 2.11"
```

Response

Status Code	Description
200 OK	Service instance was deleted. The expected response body is <code>{}</code> .
202 Accepted	Service instance deletion is in progress. This triggers the marketplace to poll the Service Instance Last Operation Endpoint for operation status.
410 Gone	Should be returned if the service instance does not exist. The expected response body is <code>{}</code> .
422 Unprocessable Entity	Should be returned if the broker only supports asynchronous deprovisioning for the requested plan and the request did not include <code>?accepts_incomplete=true</code> . The expected response body is: <code>{ "error": "AsyncRequired", "description": "This service plan requires client support for asynchronous service operations." }</code> , as described below.

Responses with any other status code will be interpreted as a failure and the service instance will remain in the marketplace database. Brokers can include a user-facing message in the `description` field; for details see [Broker Errors](#).

Body

All response bodies must be a valid JSON Object (`{}`). This is for future compatibility; it will be easier to add fields in the future if JSON is expected rather than to support the cases when a JSON body may or may not be returned.

For success responses, the following fields are supported. Others will be ignored. For error responses, see [Broker Errors](#).

Response field	Type	Description
operation	string	For asynchronous responses, service brokers may return an identifier representing the operation. The value of this field should be provided by the broker client with requests to the Last Operation endpoint in a URL encoded query parameter.

* Fields with an asterisk are required.

```
{
  "operation": "task_10"
}
```

Broker Errors

Broker failures beyond the scope of the well-defined HTTP response codes listed above (like 410 on delete) should return an appropriate HTTP response code (chosen to accurately reflect the nature of the failure) and a body containing a valid JSON Object (not an array).

Body

All response bodies must be a valid JSON Object (`{}`). This is for future compatibility; it will be easier to add fields in the future if JSON is expected rather than to support the cases when a JSON body may or may not be returned.

For error responses, the following fields are valid. Others will be ignored. If an empty JSON object is returned in the body `{}`, a generic message containing the HTTP response code returned by the broker will be displayed to the requester.

Response Field	Type	Description

Description	Response Field	Type	Description
{ "description": "Your account has exceeded its quota for service instances. Please contact support at http://support.example.com." }			

Orphans

The platform marketplace is the source of truth for service instances and bindings. Service brokers are expected to have successfully provisioned all the instances and bindings that the marketplace knows about, and none that it doesn't.

Orphans can result if the broker does not return a response before a request from the marketplace times out (typically 60 seconds). For example, if a broker does not return a response to a provision request before the request times out, the broker might eventually succeed in provisioning an instance after the marketplace considers the request a failure. This results in an orphan instance on the broker's side.

To mitigate orphan instances and bindings, the marketplace should attempt to delete resources it cannot be sure were successfully created, and should keep trying to delete them until the broker responds with a success.

Platforms should initiate orphan mitigation in the following scenarios:

Status code of broker response	Platform interpretation of response	Platform initiates orphan mitigation?
200	Success	No
200 with malformed response	Failure	No
201	Success	No
201 with malformed response	Failure	Yes
All other 2xx	Failure	Yes
408	Failure due to timeout	Yes
All other 4xx	Broker rejected request	No
5xx	Broker error	Yes
Timeout	Failure	Yes

If the platform marketplace encounters an internal error provisioning an instance or binding (for example, saving to the database fails), then it should at least send a single delete or unbind request to the service broker to prevent creation of an orphan.

Service Broker API Release Notes

Service Broker API Release Notes

v2.11

2016-11-15

- Added field `bindable` to plan objects in `/v2/catalog` response. This allows services to have both bindable and non-bindable plans.

v2.10

2016-08-01

- Service bind responses now include an optional field called `volume_mounts`. Backward incompatible changes to `volume_mounts` field in service bind response from experimental 2.9 format to final format.

v2.9

2016-06-14

- `last_operation` endpoint now supports `service_id` and `plan_id` as request parameters.
- A new field `operation` may now be returned by brokers in asynchronous responses for Provision, Update, Deprovision. This field enables brokers to provide an internal identifier for the operation that clients should provide back to the service broker when polling the `last_operation` endpoint.

v2.8

2015-11-8

- In support for Route Services, service broker may now return a `route_service_url` in the response for a create binding request.
- A broker must specify `requires: ["route_forwarding"]` in its catalog endpoint if it supports Route Services.
- Clients may now send a new field `bind_resource` with the bind request, under which the parameters required for the binding are found. This would include, for example, `app_guid` for an app binding and `route` for a route binding. For backwards compatibility, `app_guid` will remain a top-level key in addition to being included in the `bind_resource`.

v2.7

2015-10-08

- Added support for Asynchronous Operations. Brokers may now return a 202 Accepted in response to provision, update, or deprovision requests to indicate the requested operation is in progress.
- The parameter `accepts_incomplete=true` must be passed by the broker client with requests for provision, update, or deprovision to indicate support for an asynchronous response. The broker can then choose to execute the request synchronously or asynchronously.
- Added support for querying `last_operation` status at a new endpoint: `GET /v2/service_instances/:guid/last_operation`

v2.6

2015-07-23

- `app_guid` field is no longer guaranteed to be included with create service binding requests

- New field `service_id` is required with update service instance requests

v2.5

2015-06-23

- Added support for Arbitrary Parameters: service-specific configuration parameters that can be included with provision, update and bind requests

v2.4

2014-10-31

- Added support for broker clients to change the service plan for a specified service instance using new Update Service Instance endpoint

v2.3

2014-04-23

- Added `dashboard_client` field to /v2/catalog to enable broker client to provision OAuth client for a service dashboard

v2.2

2014-03-31

- Added field `free` for service plan in catalog endpoint

v2.1

2013-12-27

- New field `app_guid` is required with bind requests

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, *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 <code>unique_id</code> 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 <code>unique_id</code> 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"]}]

Access Control

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: p-riakcs
  service plan    access  orgs
  p-riakcs   developer  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-riakcs` service plans:

```
$ cf enable-service-access p-riakcs
Enabling access to all plans of service p-riakcs for all orgs as admin...
OK

$ cf service-access
getting service access as admin...
broker: p-riakcs
  service plan    access orgs
  p-riakcs   developer  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-riakcs` service plans:

```
$ cf disable-service-access p-riakcs
Disabling access to all plans of service p-riakcs for all orgs as admin...
OK

$ cf service-access
getting service access as admin...
broker: p-riakcs
  service plan    access orgs
  p-riakcs   developer  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.

Catalog Metadata

Page last updated:

The Services Marketplace is defined as the aggregate catalog of services and plans exposed to end users of a Cloud Foundry instance. Marketplace services may come from one or many service brokers. The Marketplace is exposed to end users by cloud controller clients (web, CLI, IDEs, etc), and the Cloud Foundry community is welcome to develop their own clients. All clients are not expected to have the same requirements for information to expose about services and plans. This document discusses user-facing metadata for services and plans, and how the broker API enables broker authors to provide metadata required by different cloud controller clients.

As described in the [Service Broker API](#), the only required user-facing fields are `label` and `description` for services, and `name` and `description` for service plans. Rather than attempt to anticipate all potential fields that clients will want, or add endless fields to the API spec over time, the broker API provides a mechanism for brokers to advertise any fields a client requires. This mechanism is the `metadata` field.

The contents of the `metadata` field are not validated by cloud controller but may be used by cloud controller clients. Not all clients will make use of the value of `metadata`, and not all brokers have to provide it. If a broker does advertise the `metadata` field, client developers can choose to display some or all fields available.

Community-Driven Standards

This page provides a place to publish the metadata fields required by popular cloud controller clients. Client authors can add their metadata requirements to this document, so that broker authors can see what metadata they should advertise in their catalogs.

Before adding new fields, consider whether an existing one will suffice.

 **Note:** “CLI strings” are all lowercase, no spaces. Keep it short; imagine someone having to type it as an argument for a longer CLI command.

Services Metadata Fields

Broker API Field	Type	Description	CC API Field	Pivotal CLI	Pivotal Apps Manager
name	CLI string	A short name for the service to be displayed in a catalog.	label	X	X
description	string	A short 1-line description for the service, usually a single sentence or phrase.	description	X	X
metadata.displayName	string	The name of the service to be displayed in graphical clients	extra.displayName		X
metadata.imageUrl	string	The URL to an image.	extra.imageUrl		X
metadata.longDescription	string	Long description	extra.longDescription		X
metadata.providerDisplayName	string	The name of the upstream entity providing the actual service	extra.providerDisplayName		X
metadata.documentationUrl	string	Link to documentation page for service	extra.documentationUrl		X
metadata.supportUrl	string	Link to support for the service	extra.supportUrl		X

Plan Metadata Fields

Broker API Field	Type	Description	CC API Field	Pivotal CLI	Pivotal Apps Manager
name	CLI string	A short name for the service plan to be displayed in a catalog.	name	X	
description	string	A description of the service plan to be displayed in a catalog.	description		
	array-				

Broker API Field	of-Type String	Description	extra.bullets CC API Field	Pivotal CLI	Rivotal Apps Manager
metadata.costs	cost object	An array-of-objects that describes the costs of a service, in what currency, and the unit of measure. If there are multiple costs, all of them could be billed to the user (such as a monthly + usage costs at once). Each object must provide the following keys: <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">amount: { usd: float }, unit: string</div> This indicates the cost in USD of the service plan, and how frequently the cost is occurred, such as "MONTHLY" or "per 1000 messages".	extra.costs		X
metadata.displayName	string	Name of the plan to be display in graphical clients.	extra.displayName		X

Example Broker Response Body

The example below contains a catalog of one service, having one service plan. Of course, a broker can offering a catalog of many services, each having many plans.

```
{
  "services": [
    {
      "id": "766fa866-a950-4b12-adff-c11fa4cf8fdc",
      "name": "cloudamqp",
      "description": "Managed HA RabbitMQ servers in the cloud",
      "requires": [
        ],
      "tags": [
        "amqp",
        "rabbitmq",
        "messaging"
      ],
      "metadata": {
        "displayName": "CloudAMQP",
        "imageUrl": "https://d33na3n16eqf5j.cloudfront.net/app_resources/18492/thumbs_112/img9069612145282015279.png",
        "longDescription": "Managed, highly available, RabbitMQ clusters in the cloud",
        "providerDisplayName": "84codes AB",
        "documentationUrl": "http://docs.cloudfoundry.com/docs/dotcom/marketplace/services/cloudamqp.html",
        "supportUrl": "http://www.cloudamqp.com/support.html"
      },
      "dashboard_client": {
        "id": "p-mysql-client",
        "secret": "p-mysql-secret",
        "redirect_uri": "http://p-mysql.example.com/auth/create"
      },
      "plans": [
        {
          "id": "024f3452-67f8-40bc-a724-a20c4ea24b1c",
          "name": "bunny",
          "description": "A mid-sized plan",
          "metadata": {
            "bullets": [
              "20 GB of messages",
              "20 connections"
            ],
            "costs": [
              {
                "amount": {
                  "usd": "99.0"
                },
                "unit": "MONTHLY"
              },
              {
                "amount": {
                  "usd": "0.99"
                },
                "unit": "1GB of messages over 20GB"
              }
            ],
            "displayName": "Big Bunny"
          }
        }
      ]
    }
  ]
}
```

Example Cloud Controller Response Body

```
{
  "metadata": {
    "guid": "bc8748f1-fe05-444d-ab7e-9798e1f9aef6",
    "url": "/v2/services/bc8748f1-fe05-444d-ab7e-9798e1f9aef6",
    "created_at": "2014-01-08T18:52:16+00:00",
    "updated_at": "2014-01-09T03:19:16+00:00"
  },
  "entity": {
    "label": "cloudamqp",
    "provider": "cloudamqp",
    "url": "http://adgw.a1.cf-app.example.com",
    "description": "Managed HA RabbitMQ servers in the cloud",
    "long_description": null,
    "version": "n/a",
    "info_url": null,
    "active": true,
    "bindable": true,
    "unique_id": "18723",
    "extra": {
      "displayName": "CloudAMQP",
      "imageUrl": "https://d33na3ni6eqf5j.cloudfront.net/app_resources/18723/thumbs_112/img9069612145282015279.png",
      "longDescription": "Managed, highly available, RabbitMQ clusters in the cloud",
      "providerDisplayName": "84codesAB",
      "documentationUrl": null,
      "supportUrl": null
    },
    "tags": [
      "amqp",
      "rabbitmq"
    ],
    "requires": [
    ],
    "documentation_url": null,
    "service_plans": [
      {
        "metadata": {
          "guid": "6c4903ab-14ce-41de-adb2-632cf06117a5",
          "url": "/v2/services/6c4903ab-14ce-41de-adb2-632cf06117a5",
          "created_at": "2013-11-01T00:21:25+00:00",
          "updated_at": "2014-01-09T03:19:16+00:00"
        },
        "entity": {
          "name": "bunny",
          "free": true,
          "description": "Big Bunny",
          "service_guid": "bc8748f1-fe05-444d-ab7e-9798e1f9aef6",
          "extra": {
            "bullets": [
              "20 GB of messages",
              "20 connections"
            ],
            "costs": [
              {
                "amount": {
                  "usd": 99.0
                },
                "unit": "MONTHLY"
              },
              {
                "amount": {
                  "usd": 0.99
                },
                "unit": "1GB of messages over 20GB"
              }
            ],
            "displayName": "Big Bunny"
          },
          "unique_id": "addonOffering_1889",
          "public": true
        }
      }
    ]
  }
}
```


Dashboard Single Sign-On

Page last updated:

Introduction

Single sign-on (SSO) enables Cloud Foundry users to authenticate with third-party service dashboards using their Cloud Foundry 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.

Dashboard SSO was introduced in [cf-release v169](#) so this or a newer version is required to support the feature.

Enabling the feature in Cloud Foundry

To enable the SSO feature, the Cloud Controller requires a UAA client with sufficient permissions to create and delete clients for the service brokers that request them. This client can be configured by including the following snippet in the cf-release manifest:

```
properties:  
  uaa:  
    clients:  
      cc-service-dashboards:  
        secret: cc-broker-secret  
        scope: openid,cloud_controller_service_permissions.read  
        authorities: clients.read,clients.write,clients.admin  
        authorized-grant-types: authorization_code,client_credentials
```

When this client is not present in the cf-release manifest, Cloud Controller cannot manage UAA clients and an operator will receive a warning when creating or updating service brokers that advertise the `dashboard_client` properties discussed below.

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, CF 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/`.

- 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 Cloud Foundry, 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 Cloud Foundry 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](#)).

- 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](#).
- 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.
- 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
```

- 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.
- 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 issued. 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'\n-H 'Authorization: bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJc2VyX2lkIjoid\nhttp://api.cloudfoundry.com/v2/service_instances/44b26033-1f54-4087-b7bc-da9652c2a539/permissions
```

Response:

```
{\n  \"manage\": true\n}
```

The response will indicate to the service whether this user is allowed to manage the given instance. A `true` value for the `manage` key indicates sufficient permissions; `false` would indicate insufficient permissions. Since administrators may change the permissions of users, 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
<code>openid</code>	Allows access to basic data about the user, such as email addresses
<code>cloud_controller_service_permissions.read</code>	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
<code>cloud_controller.read</code>	Allows read access to all resources the user is authorized to read
<code>cloud_controller.write</code>	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 [UAU OmniAuth Strategy](#). See this [OmniAuth wiki page](#) for instructions on how to create your own strategy.

The UAU OmniAuth strategy is used to first get an authorization code, as documented in [this section](#) of the UAU 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 repo 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 repo. 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 CL* 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 <code>DB-TYPE://USERNAME:PASSWORD@HOSTNAME:PORT/NAME</code> , where <code>DB-TYPE</code> 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 <code>name</code> 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:IwN6dCdZmeQD400ZPKpu1Y0aLx1he8wo@lemur.cloudamqp.com/ksvyjmiv"
      }
    }
  ],
  rediscloud: [
    {
      name: "rediscloud-1",
      label: "rediscloud",
      plan: "20mb",
      credentials: {
        uri: "amqp://vhuklnxa:9lNFxpTuJsAdTts98vQIdKHW3MojyMyV@lemur.cloudamqp.com/vhuklnxa"
      }
    }
  ]
}
```

```
    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 services marketplace. Developers interested in consuming these services can read the [Manage Application Requests with Route Services](#) topic.

Introduction

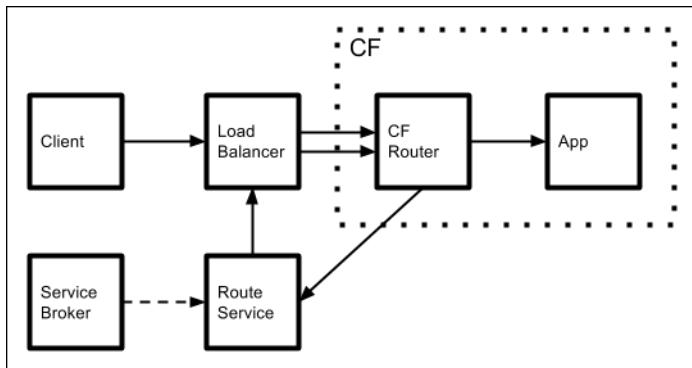
Cloud Foundry application developers may wish to apply transformation or processing to requests before they reach an application. Common examples of use cases are authentication, rate limiting, and caching services. Route Services are a new kind of Marketplace Service that developers can use to apply 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.

Architecture

Cloud Foundry supports three models for Route Services: fully-brokered services; static, brokered services; and user-provided services. In each case, you configure a route service to process traffic addressed to an app.

Fully-Brokered Service

In this 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]
```

Developers configure the service either through the service provider's web interface or by passing [arbitrary parameters](#) to their

call, through the `-c` flag.

```
cf create-service
```

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; requests for those apps do not pass through the route service.

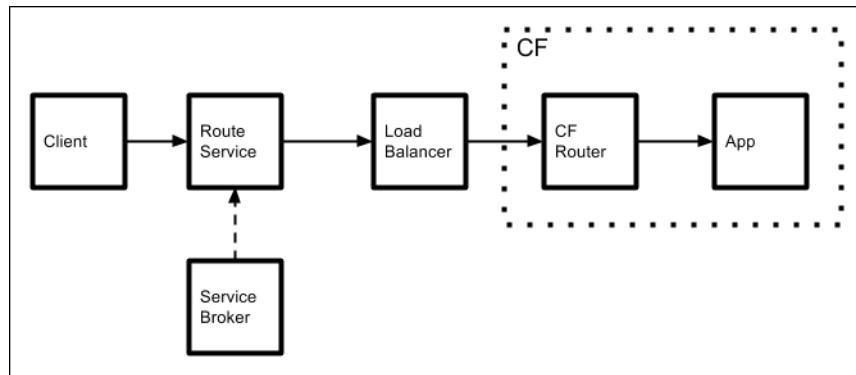
Disadvantages:

- Traffic to apps that use the route service makes additional network hops, as compared to the static model.

Static, Brokered Service

In this 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:

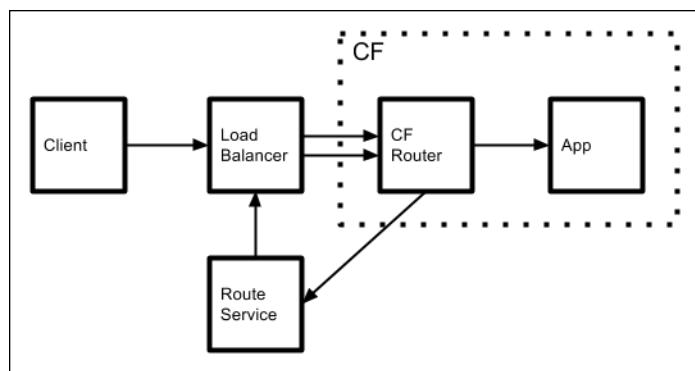
- Developers can use a Service Broker to dynamically configure how the route service processes traffic to specific applications.
- Traffic to apps that use the route service takes fewer network hops.

Disadvantages:

- Adding route services requires manual infrastructure configuration.
- Unnecessary network hops for traffic to apps that do not use the route service; 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; requests for those apps do not pass through the route service.

Disadvantages:

- Developers must manually provision/configure route services out of the context of Cloud Foundry; 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 below, as summarized in the following table:

Route Services Architecture	Fulfils CF Service Instance Responsibilities	Fulfils 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 Elastic Runtime tile, under [Settings > Networking](#). Depending on your infrastructure, refer to the Elastic Runtime configuration topics for [Amazon Web Services](#), [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 will associate the `route_service_url` with the route in the Cloud Foundry router. All requests for the route will be proxied to the URL specified by `route_service_url`.

Once a route service completes its function, it is expected to forward the request to the route the original request was sent to. The Cloud Foundry router will include a header that provides the address of the route, as well as two headers that are used by the route itself to validate the request sent by the route service.

Headers

The `X-CF-Forwarded-Url` header contains the URL of the application route. The route service should forward the request to this URL.

The route service should not strip off the `X-CF-Proxy-Signature` and `X-CF-Proxy-Metadata`, as the Gorouter relies on these headers to validate the request.

SSL Certificates

When Cloud Foundry is deployed in a development environment, certificates hosted by the load balancer will be self-signed (not signed by a trusted certificate authority). When the route service has finished 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 the number of seconds configured by the `router.route_service_timeout` property (default 60 seconds).

In addition, all requests must respond in the number of seconds configured by the `request_timeout_in_seconds` property (default 900 seconds).

Timeouts are configurable for the router using the cf-release BOSH deployment manifest. For more information, see the [spec](#).

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 will send 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 a `https` scheme, otherwise the Cloud Controller will reject the binding. `route_service_url` is optional; 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 Cloud Foundry 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](#).

Requires CLI version 6.16 or above.

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 will create 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 see 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.

Logging and Metrics

Loggregator is the next generation system for aggregating and streaming logs and metrics from all of the user apps and system components in Elastic Runtime.

Table of Contents

- [Overview of the Loggregator System](#)
- [Loggregator Guide for Cloud Foundry Operators](#)
- [Application Logging in Cloud Foundry](#)
- [Security Event Logging for Cloud Controller and UAA](#)
- [Cloud Foundry Component Metrics](#)
- [Deploying a Nozzle to the Loggregator Firehose](#)
- [Cloud Foundry Data Sources](#)
- [Installing the Loggregator Plugin for cf CLI](#)
- [Monitoring a Pivotal Cloud Foundry Deployment](#)
- [Using SSL with a Self-Signed Certificate in JMX Bridge](#) 
- [Deploying JMX Bridge](#) 
- [Using JMX Bridge](#) 

Overview of the Loggregator System

Page last updated:

Loggregator is the next generation system for aggregating and streaming logs and metrics from all of the user apps and system components in an Elastic Runtime deployment.

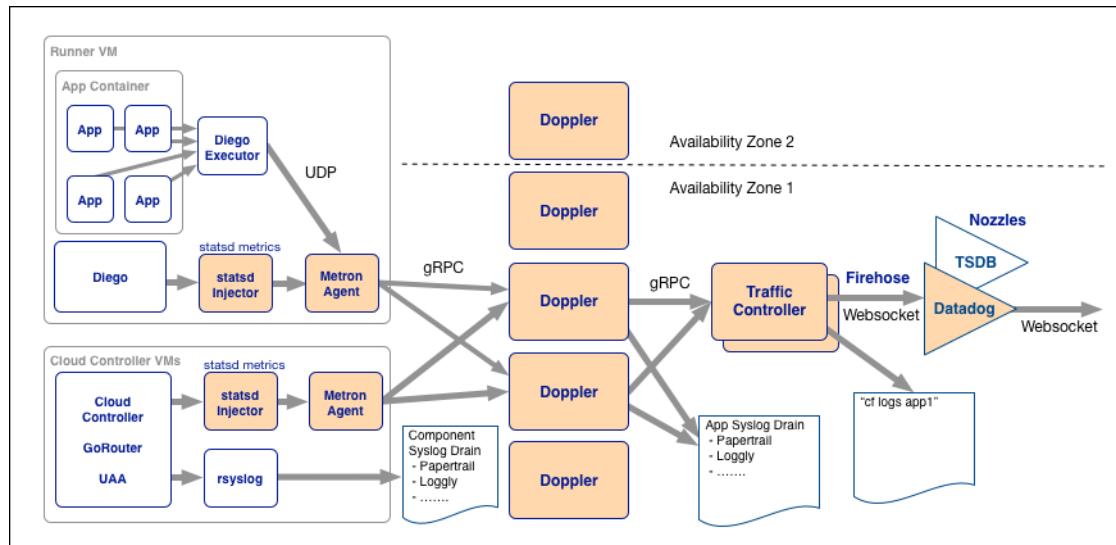
View the [Loggregator repository on GitHub](#).

Using Loggregator

The main use cases are as follows:

- 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 listens to the Firehose for specified events and metrics, and streams this data to external services.

Loggregator Components



To see a larger version of this diagram, click [here](#).

Note: The Loggregator system now 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 Doppers.

Source

Sources are logging agents that run on the Cloud Foundry components.

Metron

Metron agents are co-located 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

Handles client requests for logs. Gathers and collates messages from all Doppler servers, and provides external API and message translation (as needed for legacy APIs). Exposes the Firehose.

Firehose

The Firehose is a WebSocket endpoint which streams all the event data coming from an Elastic Runtime deployment. The data stream includes logs, HTTP events and container metrics from all applications, and metrics from all Elastic Runtime system components. Logs from system components such as Cloud Controller are not included in the firehose and are typically accessed via rsyslog configuration.

Because the data coming from the Firehose may contain sensitive information, such as customer information in the application logs, the Firehose is only accessible by users who have the right permissions.

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](#).

The address of the traffic controller can be discovered by hitting the `/info` endpoint on the API and getting the value of the `doppler_logging_endpoint`.

Example output 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 or 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

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. For example:

- The JMX Bridge OpenTSDB Firehose Nozzle, which installs with [JMX Bridge](#)
- The [Datadog nozzle](#) publishes metrics coming from the Firehose to Datadog.
- The [Syslog nozzle](#) filters out log messages coming from the Firehose and sends it to a syslog server.

See our [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.

Scaling Loggregator

When the volume of log and metric data generated by Elastic Runtime components exceeds the storage buffer capacity of the Dopplers that collect it, data can be lost. [Configuring System Logging in Elastic Runtime](#) explains how to scale the Loggregator system to keep up with high stream volume and minimize data loss.

Scaling Nozzles

You can scale [nozzles](#) 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 events across all instances of the nozzle. For example, if you have two nozzles with the same subscription ID, the Firehose sends half of the events to one nozzle and half to the other. Similarly, if you have three nozzles with the same subscription ID, the Firehose sends each instance one-third of the event traffic.

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 `TruncatingBuffer.DroppedMessages`. The nozzle receives both messages from the Firehose, alerting the operator to the performance issue.
- **PolicyViolation** error: The Traffic Controller periodically sends `ping` control messages over the Firehose WebSocket connection. If a client does not respond to a `ping` with a `pong` message within 30 seconds, the Traffic Controller closes the WebSocket connection with the WebSocket error code `ClosePolicyViolation (1008)`. The nozzle should intercept this WebSocket close error, alerting the operator to the performance issue.

An operator can scale the number of nozzles in response to these alerts to minimize the loss of data.

Forwarding Logs to an External Service

You can configure Elastic Runtime to forward log data from components and apps to an external aggregator service instead of routing it to the Loggregator Firehose. [Configuring System Logging in Elastic Runtime](#) explains how to enable log forwarding by specifying the aggregator address, port, and protocol.

[Using Log Management Services](#) explains how to bind applications to the external service and configure it to receive logs from Elastic Runtime.

Log Message Size Constraints

The Diego cell emits application logs as UDP messages to the Metron. Diego breaks up log messages greater than approximately 60KiB into multiple envelopes to mitigate this constraint.

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 `STDOUT` or `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 insures 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 or DEA 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
```

DEA

The Droplet Execution Agent emits DEA logs beginning when it starts or stops the app. These actions implement the desired state requested by the user. The DEA also emits messages when an app crashes.

For example:

```
2014-02-13T11:44:52.07-0800 [DEA] OUT Starting app instance (index 1) with guid e1ca6390-cf78-4fc7-9d86-5b7ed01e9c28
```

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 RTR` 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": "jse
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 the purposes of security or compliance.

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 Elastic Runtime](#) 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 user=USERNAME uid=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 above:

- `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= uid= 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 uid=
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 uid=c7ca208f-8a9e-4aab-
92f5-28795f86d62a request=/v2/routes?inline-relations-depth=1&q=
host%3Adora%3Bdomain_guid%3B777-109f-5fn-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-aface-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 uid=a0012026-55ic-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-d59cf7cb7ea
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 uid=49a33f9-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 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 Elastic Runtime](#) 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-cef2-838abcd0i3e3,
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.

Cloud Foundry Component Metrics

Page last updated:

This topic lists and describes the metrics available for Pivotal Cloud Foundry (PCF) system components. These metrics are streamed from the Loggregator [Firehose](#).

Cloud Controller

Default Origin Name: cc

Metric Name	Description
failed_job_count.<VM_NAME>-<VM_INDEX>	Number of failed jobs in the <VM_NAME>-<VM_INDEX> queue. This is the number of delayed jobs where the <code>failed_at</code> column is populated with the time of the most recently failed attempt at the job. The failed job count is not specific to the jobs run by the Cloud Controller worker. By default, Cloud Controller deletes failed jobs after 31 days. Emitted every 30 seconds per VM.
failed_job_count.cc-generic	Number of failed jobs in the cc-generic queue. By default, Cloud Controller deletes failed jobs after 31 days. Emitted every 30 seconds per VM.
failed_job_count.total	Number of failed jobs in all queues. By default, Cloud Controller deletes failed jobs after 31 days. Emitted every 30 seconds per VM.
http_status.1XX	Number of HTTP response status codes of type 1xx (informational). This resets when the Cloud Controller process is restarted and is incremented at the end of each request cycle.
http_status.2XX	Number of HTTP response status codes of type 2xx (success). This resets when the Cloud Controller process is restarted and is incremented at the end of each request cycle. Emitted for each Cloud Controller request.
http_status.3XX	Number of HTTP response status codes of type 3xx (redirection). This resets when the Cloud Controller process is restarted and is incremented at the end of each request cycle. Emitted for each Cloud Controller request.
http_status.4XX	Number of HTTP response status codes of type 4xx (client error). This resets when the Cloud Controller process is restarted and is incremented at the end of each request cycle. Emitted for each Cloud Controller request.
http_status.5XX	Number of HTTP response status codes of type 5xx (server error). This resets when the Cloud Controller process is restarted and is incremented at the end of each request cycle.
job_queue_length.cc-<VM_NAME>-<VM_INDEX>	Number of background jobs in the <VM_NAME>-<VM_INDEX> queue that have yet to run for the first time. Emitted every 30 seconds per VM.
job_queue_length.cc-generic	Number of background jobs in the cc-generic queue that have yet to run for the first time. Emitted every 30 seconds per VM.
job_queue_length.total	Total number of background jobs in the queues that have yet to run for the first time. Emitted every 30 seconds per VM.
log_count.all	Total number of log messages, sum of messages of all severity levels. The count resets when the Cloud Controller process is restarted. Emitted every 30 seconds per VM.
log_count.debug	Number of log messages of severity “debug.” The count resets when the Cloud Controller process is restarted. Emitted every 30 seconds per VM.
log_count.debug1	Not used.
log_count.debug2	Number of log messages of severity “debug2.” The count resets when the Cloud Controller process is restarted. Emitted every 30 seconds per VM.
log_count.error	Number of log messages of severity “error.” Error is the most severe level. It is used for failures and during error handling. Most errors can be found under this log level, eg. failed unbinding a service, failed to cancel a task, Diego app crashed error, staging completion errors, staging errors, and resource not found. The count resets when the Cloud Controller process is restarted. Emitted every 30 seconds per VM.
log_count.fatal	Number of log messages of severity “fatal.” The count resets when the Cloud Controller process is restarted. Emitted every 30 seconds per VM.

Metric Name	Description
log_count.info	Number of log messages of severity "info." Examples of info messages are droplet created, copying package, uploading package, access denied due to insufficient scope, job logging, blobstore actions, staging requests, and app running requests. The count resets when the Cloud Controller process is restarted. Emitted every 30 seconds per VM.
log_count.off	Number of log messages of severity "off." The count resets when the Cloud Controller process is restarted. Emitted every 30 seconds per VM.
log_count.warn	Number of log messages of severity "warn." Warn is also used for failures and during error handling, eg. diagnostics written to file, failed to capture diagnostics, app rollback failed, service broker already deleted, and UAA token problems. The count resets when the Cloud Controller process is restarted. Emitted every 30 seconds per VM.
requests.completed	Number of requests that have been processed. Emitted for each Cloud Controller request.
requests.outstanding	Number of request that are currently being processed. Emitted for each Cloud Controller request.
tasks_running.count	Number of currently running tasks. Emitted every 30 seconds per VM. This metric is only seen in version 3 of the Cloud Foundry API.
tasks_running.memory_in_mb	Memory being consumed by all currently running tasks. Emitted every 30 seconds per VM. This metric is only seen in version 3 of the Cloud Foundry API.
thread_info.event_machine.connection_count	Number of open connections to event machine. Emitted every 30 seconds per VM.
thread_info.event_machine.resultqueue.num_waiting	Number of scheduled tasks in the result. Emitted every 30 seconds per VM.
thread_info.event_machine.resultqueue.size	Number of unscheduled tasks in the result. Emitted every 30 seconds per VM.
thread_info.event_machine.threadqueue.num_waiting	Number of scheduled tasks in the threadqueue. Emitted every 30 seconds per VM.
thread_info.event_machine.threadqueue.size	Number of unscheduled tasks in the threadqueue. Emitted every 30 seconds per VM.
thread_info.thread_count	Total number of threads that are either runnable or stopped. Emitted every 30 seconds per VM.
total_users	Total number of users ever created, including inactive users. Emitted every 10 minutes per VM.
vcap_sinatra.recent_errors	50 most recent errors. DEPRECATED
vitals.cpu	Percentage of CPU used by the Cloud Controller process. Emitted every 30 seconds per VM.
vitals.cpu_load_avg	System CPU load averaged over the last 1 minute according to the OS. Emitted every 30 seconds per VM.
vitals.mem_bytes	The RSS bytes (resident set size) or real memory of the Cloud Controller process. Emitted every 30 seconds per VM.
vitals.mem_free_bytes	Total memory available according to the OS. Emitted every 30 seconds per VM.
vitals.mem_used_bytes	Total memory used (active + wired) according to the OS. Emitted every 30 seconds per VM.
vitals.num_cores	The number of CPUs of a host machine. Emitted every 30 seconds per VM.
vitals.uptime	The uptime of the Cloud Controller process in seconds. Emitted every 30 seconds per VM.

[Top](#)

Diego

Diego metrics have the following origin names:

- [auctioneer](#)
- [bbs](#)
- [cc_uploader](#)
- [file_server](#)
- [garden_linux](#)

- [nsync_bulker](#)
- [nsync_listener](#)
- [rep](#)
- [route_emitter](#)
- [ssh_proxy](#)
- [stager](#)
- [tps_listener](#)
- [tps_watcher](#)

Default Origin Name: auctioneer

Metric Name	Description
AuctioneerFetchStatesDuration	Time in nanoseconds that the auctioneer took to fetch state from all the cells when running its auction. Emitted every 30 seconds during each auction.
AuctioneerLRPAuctionsFailed	Cumulative number of LRP instances that the auctioneer failed to place on Diego cells. Emitted every 30 seconds during each auction.
AuctioneerLRPAuctionsStarted	Cumulative number of LRP instances that the auctioneer successfully placed on Diego cells. Emitted every 30 seconds during each auction.
AuctioneerTaskAuctionsFailed	Cumulative number of Tasks that the auctioneer failed to place on Diego cells. Emitted every 30 seconds during each auction.
AuctioneerTaskAuctionsStarted	Cumulative number of Tasks that the auctioneer successfully placed on Diego cells. Emitted every 30 seconds during each auction.
LockHeld.v1-locks-auctioneer_lock	Whether an auctioneer holds the auctioneer lock: <code>1</code> means the lock is held, and <code>0</code> means the lock was lost. Emitted every 30 seconds by the active auctioneer.
LockHeldDuration.v1-locks-auctioneer_lock	Time in nanoseconds that the active auctioneer has held the auctioneer lock. Emitted every 30 seconds by the active auctioneer.
memoryStats.lastGCPauseTimeNS	Duration in nanoseconds of the last garbage collector pause.
memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use.
memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use.
memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator.
memoryStats.numFrees	Lifetime number of memory deallocations.
memoryStats.numMallocs	Lifetime number of memory allocations.
numCPUS	Number of CPUs on the machine.
numGoRoutines	Instantaneous number of active goroutines in the process.

Default Origin Name: bbs

Metric Name	Description
BBSMasterElected	Emitted once when the BBS is elected as master.
ConvergenceLRPDuration	Time in nanoseconds that the BBS took to run its LRP convergence pass. Emitted every 30 seconds when LRP convergence runs.
ConvergenceLRPPreProcessingActualLRPsDeleted	Cumulative number of times the BBS has detected and deleted a malformed ActualLRP in its LRP convergence pass. Emitted every 30 seconds.
ConvergenceLRPPreProcessingMalformedRunInfos	Cumulative number of times the BBS has detected a malformed DesiredLRP RunInfo in its LRP convergence pass. Emitted every 30 seconds.
ConvergenceLRPPreProcessingMalformedSchedulingInfos	Cumulative number of times the BBS has detected a malformed DesiredLRP SchedulingInfo in its LRP convergence pass. Emitted every 30 seconds.
ConvergenceLPRRuns	Cumulative number of times BBS has run its LRP convergence pass. Emitted every 30 seconds.
ConvergenceTaskDuration	Time in nanoseconds that the BBS took to run its Task convergence pass. Emitted every 30 seconds when Task convergence runs.
ConvergenceTaskRuns	Cumulative number of times the BBS has run its Task convergence pass. Emitted every 30 seconds.
ConvergenceTasksKicked	Cumulative number of times the BBS has updated a Task during its Task

Metric Name	Description
ConvergenceTasksPruned	Cumulative number of times the BBS has deleted a malformed Task during its Task convergence pass. Emitted every 30 seconds.
CrashedActualLRPs	Total number of LRP instances that have crashed. Emitted every 30 seconds.
CrashingDesiredLRPs	Total number of DesiredLRPs that have at least one crashed instance. Emitted every 30 seconds.
Domain.cf-apps	Whether the 'cf-apps' domain is up-to-date, so that CF apps from CC have been synchronized with DesiredLRPs for Diego to run. <code>1</code> means the domain is up-to-date, no data means it is not. Emitted every 30 seconds.
Domain.cf-tasks	Whether the 'cf-tasks' domain is up-to-date, so that CF tasks from CC have been synchronized with tasks for Diego to run. <code>1</code> means the domain is up-to-date, no data means it is not. Emitted every 30 seconds.
ETCDLeader	Index of the leader node in the etcd cluster. Emitted every 30 seconds.
ETCDRaftTerm	Raft term of the etcd cluster. Emitted every 30 seconds.
ETCDReceivedBandwidthRate	Number of bytes per second received by the follower etcd node. Emitted every 30 seconds.
ETCDReceivedRequestRate	Number of requests per second received by the follower etcd node. Emitted every 30 seconds.
ETCDSentBandwidthRate	Number of bytes per second sent by the leader etcd node. Emitted every 30 seconds.
ETCDSentRequestRate	Number of requests per second sent by the leader etcd node. Emitted every 30 seconds.
ETCDWatchers	Number of watches set against the etcd cluster. Emitted every 30 seconds.
LockHeld.v1-locks-bbs_lock	Whether a BBS holds the BBS lock: <code>1</code> means the lock is held, and <code>0</code> means the lock was lost. Emitted every 30 seconds by the active BBS server.
LockHeldDuration.v1-locks-bbs_lock	Time in nanoseconds that the active BBS has held the BBS lock. Emitted every 30 seconds by the active BBS server.
LRPsClaimed	Total number of LRP instances that have been claimed by some cell. Emitted every 30 seconds.
LRPsDesired	Total number of LRP instances desired across all LRPs. Emitted periodically.
LRPsExtra	Total number of LRP instances that are no longer desired but still have a BBS record. Emitted every 30 seconds.
LRPsMissing	Total number of LRP instances that are desired but have no record in the BBS. Emitted every 30 seconds.
LRPsRunning	Total number of LRP instances that are running on cells. Emitted every 30 seconds.
LRPsUnclaimed	Total number of LRP instances that have not yet been claimed by a cell. Emitted every 30 seconds.
memoryStats.lastGCPauseTimeNS	Duration in nanoseconds of the last garbage collector pause.
memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use.
memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use.
memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator.
memoryStats.numFrees	Lifetime number of memory deallocations.
memoryStats.numMallocs	Lifetime number of memory allocations.
MetricsReportingDuration	Time in nanoseconds that the BBS took to emit metrics about etcd. Emitted every 30 seconds.
MigrationDuration	Time in nanoseconds that the BBS took to run migrations against its persistence store. Emitted each time a BBS becomes the active master.
numCPUs	Number of CPUs on the machine.
numGoRoutines	Instantaneous number of active goroutines in the process.
RequestCount	Cumulative number of requests the BBS has handled through its API. Emitted for each BBS request.
RequestLatency	Time in nanoseconds that the BBS took to handle requests to its API endpoints.

Metric Name	Description
TasksCompleted	Emitted when the BBS API handles requests. Total number of Tasks that have completed. Emitted every 30 seconds.
TasksPending	Total number of Tasks that have not yet been placed on a cell. Emitted every 30 seconds.
TasksResolving	Total number of Tasks locked for deletion. Emitted every 30 seconds.
TasksRunning	Total number of Tasks running on cells. Emitted every 30 seconds.

Default Origin Name: cc_uploader

Metric Name	Description
memoryStats.lastGCPauseTimeNS	Duration in nanoseconds of the last garbage collector pause.
memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use.
memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use.
memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator.
memoryStats.numFrees	Lifetime number of memory deallocations.
memoryStats.numMallocs	Lifetime number of memory allocations.
numCPUS	Number of CPUs on the machine.
numGoRoutines	Instantaneous number of active goroutines in the process.

Default Origin Name: file_server

Metric Name	Description
memoryStats.lastGCPauseTimeNS	Duration in nanoseconds of the last garbage collector pause.
memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use.
memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use.
memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator.
memoryStats.numFrees	Lifetime number of memory deallocations.
memoryStats.numMallocs	Lifetime number of memory allocations.
numCPUS	Number of CPUs on the machine.
numGoRoutines	Instantaneous number of active goroutines in the process.

Default Origin Name: garden_linux

Metric Name	Description
BackingStores	Number of container backing store files. Emitted every 30 seconds.
DepotDirs	Number of directories in the Garden depot. Emitted every 30 seconds.
LoopDevices	Number of attached loop devices. Emitted every 30 seconds.
memoryStats.lastGCPauseTimeNS	Duration in nanoseconds of the last garbage collector pause.
memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use.
memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use.
memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator.
memoryStats.numFrees	Lifetime number of memory deallocations.
memoryStats.numMallocs	Lifetime number of memory allocations.
MetricsReporting	How long it took to emit the BackingStores, DepotDirs, and LoopDevices metrics. Emitted every 30 seconds.
numCPUS	Number of CPUs on the machine.
numGoRoutines	Instantaneous number of active goroutines in the process.

Default Origin Name: nsync_bulker

Metric Name	Description
DesiredLRPSyncDuration	Time in nanoseconds that the nsync-bulker took to synchronize CF apps and Diego DesiredLRPs. Emitted every 30 seconds.
LockHeld.v1-locks-nsync_bulker_lock	Whether an nsync-bulker holds the nsync-bulker lock: <code>1</code> means the lock is held, and <code>0</code> means the

Metric Name	Description
LockHeldDuration.v1-locks-nsync_bulker_lock	Time in nanoseconds that the active nsync-bulker has held the convergence lock. Emitted every 30 seconds by the active nsync-bulker.
LRPsDesired	Cumulative number of LRPCs desired through the nsync API. Emitted on each request desiring a new LRP, every 30 seconds.
memoryStats.lastGCPauseTimeNS	Duration in nanoseconds of the last garbage collector pause.
memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use.
memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use.
memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator.
memoryStats.numFrees	Lifetime number of memory deallocations.
memoryStats.numMallocs	Lifetime number of memory allocations.
NsyncInvalidDesiredLRPsFound	Number of invalid DesiredLRPs found during nsync-bulker periodic synchronization. Emitted every 30 seconds.
numCPUS	Number of CPUs on the machine.
numGoRoutines	Instantaneous number of active goroutines in the process.

Default Origin Name: nsync_listener

Metric Name	Description
memoryStats.lastGCPauseTimeNS	Duration in nanoseconds of the last garbage collector pause.
memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use.
memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use.
memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator.
memoryStats.numFrees	Lifetime number of memory deallocations.
memoryStats.numMallocs	Lifetime number of memory allocations.
numCPUS	Number of CPUs on the machine.
numGoRoutines	Instantaneous number of active goroutines in the process.

Default Origin Name: rep

Metric Name	Description
CapacityRemainingContainers	Remaining number of containers this cell can host. Emitted every 30 seconds.
CapacityRemainingDisk	Remaining amount in MiB of disk available for this cell to allocate to containers. Emitted every 30 seconds.
CapacityRemainingMemory	Remaining amount in MiB of memory available for this cell to allocate to containers. Emitted every 30 seconds.
CapacityTotalContainers	Total number of containers this cell can host. Emitted every 30 seconds.
CapacityTotalDisk	Total amount in MiB of disk available for this cell to allocate to containers. Emitted every 30 seconds.
CapacityTotalMemory	Total amount in MiB of memory available for this cell to allocate to containers. Emitted every 30 seconds.
CM	Emitted every 30 seconds.
ContainerCount	Number of containers hosted on the cell. Emitted every 30 seconds.
GardenContainerCreationDuration	Time in nanoseconds that the rep Garden backend took to create a container. Emitted after every successful container creation.
LogMessage	Emitted every 30 seconds.
logSenderTotalMessagesRead	Count of application log messages sent by Diego Executor. Emitted every 30 seconds.
memoryStats.lastGCPauseTimeNS	Duration in nanoseconds of the last garbage collector pause.
memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use.
memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use.
memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator.
memoryStats.numFrees	Lifetime number of memory deallocations.

Metric Name	Description
memoryStats.numMallocs	Lifetime number of memory allocations.
numCPUS	Number of CPUs on the machine.
numGoRoutines	Instantaneous number of active goroutines in the process.
RepBulkSyncDuration	Time in nanoseconds that the cell rep took to synchronize the ActualLRPs it has claimed with its actual garden containers. Emitted every 30 seconds by each rep.
UnhealthyCell	Whether the cell has failed to pass its healthcheck against the garden backend. <code>0</code> signifies healthy, and <code>1</code> signifies unhealthy. Emitted every 30 seconds.

Default Origin Name: route_emitter

Metric Name	Description
LockHeld.v1-locks-route_emitter_lock	Whether a route-emitter holds the route-emitter lock: <code>1</code> means the lock is held, and <code>0</code> means the lock was lost. Emitted every 30 seconds by the active route-emitter.
LockHeldDuration.v1-locks-route_emitter_lock	Time in nanoseconds that the active route-emitter has held the route-emitter lock. Emitted every 30 seconds by the active route-emitter.
memoryStats.lastGCPauseTimeNS	Duration in nanoseconds of the last garbage collector pause.
memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use.
memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use.
memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator.
memoryStats.numFrees	Lifetime number of memory deallocations.
memoryStats.numMallocs	Lifetime number of memory allocations.
MessagesEmitted	The cumulative number of registration messages that this process has sent. Emitted every 30 seconds.
numCPUS	Number of CPUs on the machine.
numGoRoutines	Instantaneous number of active goroutines in the process.
RouteEmitterSyncDuration	Time in nanoseconds that the active route-emitter took to perform its synchronization pass. Emitted every 30 seconds.
RoutesRegistered	Cumulative number of route registrations emitted from the route-emitter as it reacts to changes to LRPCs. Emitted every 30 seconds.
RoutesSynced	Cumulative number of route registrations emitted from the route-emitter during its periodic routetable synchronization. Emitted every 30 seconds.
RoutesTotal	Number of routes in the route-emitter's routing table. Emitted every 30 seconds.
RoutesUnregistered	Cumulative number of route unregistrations emitted from the route-emitter as it reacts to changes to LRPCs. Emitted every 30 seconds .

Default Origin Name: ssh_proxy

Metric Name	Description
memoryStats.lastGCPauseTimeNS	Duration in nanoseconds of the last garbage collector pause.
memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use.
memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use.
memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator.
memoryStats.numFrees	Lifetime number of memory deallocations.
memoryStats.numMallocs	Lifetime number of memory allocations.
numCPUS	Number of CPUs on the machine.
numGoRoutines	Instantaneous number of active goroutines in the process .

Default Origin Name: stager

Metric Name	Description
memoryStats.lastGCPauseTimeNS	Duration in nanoseconds of the last garbage collector pause.
memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use.
memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use.
memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator.

Metric Name	Description
memoryStats.numFrees	Lifetime number of memory deallocations.
memoryStats.numMallocs	Lifetime number of memory allocations.
numCPUS	Number of CPUs on the machine.
numGoRoutines	Instantaneous number of active goroutines in the process.
StagingRequestFailedDuration	Time in nanoseconds that the failed staging task took to run. Emitted each time a staging task fails.
StagingRequestsFailed	Cumulative number of failed staging tasks handled by each stager. Emitted every time a staging task fails.
StagingRequestsSucceeded	Cumulative number of successful staging tasks handled by each stager. Emitted every time a staging task completes successfully.
StagingRequestSucceededDuration	Time in nanoseconds that the successful staging task took to run. Emitted each time a staging task completes successfully.
StagingStartRequestsReceived	Cumulative number of requests to start a staging task. Emitted by a stager each time it handles a request .

Default Origin Name: tps_listener

Metric Name	Description
memoryStats.lastGCPauseTimeNS	Duration in nanoseconds of the last garbage collector pause.
memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use.
memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use.
memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator.
memoryStats.numFrees	Lifetime number of memory deallocations.
memoryStats.numMallocs	Lifetime number of memory allocations.
numCPUS	Number of CPUs on the machine.
numGoRoutines	Instantaneous number of active goroutines in the process.

Default Origin Name: tps_watcher

Metric Name	Description
LockHeld.v1-locks-tps_watcher_lock	Whether a tps-watcher holds the tps-watcher lock: <input checked="" type="checkbox"/> means the lock is held, and <input type="checkbox"/> means the lock was lost. Emitted every 30 seconds by the active tps-watcher.
LockHeldDuration.v1-locks-tps_watcher_lock	Time in nanoseconds that the active tps-watcher has held the convergence lock. Emitted every 30 seconds by the active tps-watcher.
memoryStats.lastGCPauseTimeNS	Duration in nanoseconds of the last garbage collector pause.
memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use.
memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use.
memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator.
memoryStats.numFrees	Lifetime number of memory deallocations.
memoryStats.numMallocs	Lifetime number of memory allocations.
numCPUS	Number of CPUs on the machine. Emitted every 30 seconds.
numGoRoutines	Instantaneous number of active goroutines in the process.

[Top](#)

DopplerServer

Default Origin Name: DopplerServer

Metric Name	Description
dropsondeListener.currentBufferCount	DEPRECATED
dropsondeListener.receivedByteCount	DEPRECATED in favor of DopplerServer.udpListener.receivedByteCount.
dropsondeListener.receivedMessageCount	DEPRECATED in favor of DopplerServer.udpListener.receivedMessageCount.
dropsondeUnmarshaller.containerMetricReceived	Lifetime number of ContainerMetric messages unmarshalled.

Metric Name	Description
dropsondeUnmarshaller.counterEventReceived	Lifetime number of CounterEvent messages unmarshalled.
dropsondeUnmarshaller.errorReceived	Lifetime number of Error messages unmarshalled.
dropsondeUnmarshaller.heartbeatReceived	DEPRECATED
dropsondeUnmarshaller.httpStartStopReceived	Lifetime number of HttpStartStop messages unmarshalled.
dropsondeUnmarshaller.logMessageTotal	Lifetime number of LogMessage messages unmarshalled.
dropsondeUnmarshaller.unmarshalErrors	Lifetime number of errors when unmarshalling messages.
dropsondeUnmarshaller.valueMetricReceived	Lifetime number of ValueMetric messages unmarshalled.
httpServer.receivedMessages	Number of messages received by Doppler's internal MessageRouter. Emitted every 5 seconds.
LinuxFileDescriptor	Number of file handles for the Doppler's process.
memoryStats.lastGCPauseTimeNS	Duration of the last Garbage Collector pause in nanoseconds.
memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use.
memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use.
memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator.
memoryStats.numFrees	Lifetime number of memory deallocations.
memoryStats.numMallocs	Lifetime number of memory allocations.
messageRouter.numberOfContainerMetricSinks	Instantaneous number of container metric sinks known to the SinkManager. Emitted every 5 seconds.
messageRouter.numberOfDumpSinks	Instantaneous number of dump sinks known to the SinkManager. Emitted every 5 seconds.
messageRouter.numberOfFirehoseSinks	Instantaneous number of firehose sinks known to the SinkManager. Emitted every 5 seconds.
messageRouter.numberOfSyslogSinks	Instantaneous number of syslog sinks known to the SinkManager.
messageRouter.numberOfWebsocketSinks	Instantaneous number of WebSocket sinks known to the SinkManager. Emitted every 5 seconds.
messageRouter.totalDroppedMessages	Lifetime number of messages dropped inside Doppler for various reasons (downstream consumer can't keep up internal object wasn't ready for message, etc.).
sentMessagesFirehose.<SUBSCRIPTION_ID>	Number of sent messages through the firehose per subscription id. Emitted every 5 seconds.
udpListener.receivedByteCount	Lifetime number of bytes received by Doppler's UDP Listener.
udpListener.receivedMessageCount	Lifetime number of messages received by Doppler's UDP Listener.
udpListener.receivedErrorCount	Lifetime number of errors encountered by Doppler's UDP Listener while reading from the connection.
tcpListener.receivedByteCount	Lifetime number of bytes received by Doppler's TCP Listener. Emitted every 5 seconds.
tcpListener.receivedMessageCount	Lifetime number of messages received by Doppler's TCP Listener. Emitted every 5 seconds.
tcpListener.receivedErrorCount	Lifetime number of errors encountered by Doppler's TCP Listener while handshaking, decoding or reading from the connection.
tlsListener.receivedByteCount	Lifetime number of bytes received by Doppler's TLS Listener. Emitted every 5 seconds.
tlsListener.receivedMessageCount	Lifetime number of messages received by Doppler's TLS Listener. Emitted every 5 seconds.
tlsListener.receivedErrorCount	Lifetime number of errors encountered by Doppler's TLS Listener while handshaking, decoding or reading from the connection.
TruncatingBuffer.DroppedMessages	Number of messages intentionally dropped by Doppler from the sink for the specific sink. This counter event will correspond with log messages "Log message output is too high." Emitted every 5 seconds.
TruncatingBuffer.totalDroppedMessages	Lifetime total number of messages intentionally dropped by Doppler from all of its sinks due to back pressure. Emitted every 5 seconds.
listeners.totalReceivedMessageCount	Total number of messages received across all of Doppler's listeners (UDP, TCP, TLS).
numCpus	Number of CPUs on the machine.
numGoRoutines	Instantaneous number of active goroutines in the Doppler process.

Metric Name	Description
signatureVerifier.invalidSignatureErrors	Lifetime number of messages received with an invalid signature.
signatureVerifier.missingSignatureErrors	Lifetime number of messages received that are too small to contain a signature.
signatureVerifier.validSignatures	Lifetime number of messages received with valid signatures.
Uptime	Uptime for the Doppler's process.

[Top](#)

Etcd

Visit [etcd stats API](#) 

Default Origin Name: etcd

Metric Name	Description
CompareAndDeleteFail	CompareAndDeleteFail operation count. Emitted every 30 seconds.
CompareAndDeleteSuccess	CompareAndDeleteSuccess operation count. Emitted every 30 seconds.
CompareAndSwapFail	CompareAndSwapFail operation count. Emitted every 30 seconds.
CompareAndSwapSuccess	CompareAndSwapSuccess operation count. Emitted every 30 seconds.
CreateFail	CreateFail operation count. Emitted every 30 seconds.
CreateSuccess	CreateSuccess operation count. Emitted every 30 seconds.
DeleteFail	DeleteFail operation count. Emitted every 30 seconds.
DeleteSuccess	DeleteSuccess operation count. Emitted every 30 seconds.
EtcdIndex	X-Etcd-Index value from the /stats/store endpoint. Emitted every 30 seconds.
ExpireCount	ExpireCount operation count. Emitted every 30 seconds.
Followers	Number of etcd followers. Emitted every 30 seconds.
GetsFail	GetsFail operation count. Emitted every 30 seconds.
GetsSuccess	GetsSuccess operation count. Emitted every 30 seconds.
IsLeader	<code>1</code> if the current server is the leader, <code>0</code> if it is a follower. Emitted every 30 seconds.
Latency	Current latency in milliseconds from leader to a specific follower. Emitted every 30 seconds.
RaftIndex	X-Raft-Index value from the <code>/stats/store</code> endpoint. Emitted every 30 seconds.
RaftTerm	X-Raft-Term value from the <code>/stats/store</code> endpoint. Emitted every 30 seconds.
ReceivedAppendRequests	Number of append requests this node has processed. Emitted every 30 seconds.
ReceivingBandwidthRate	Number of bytes per second this node is receiving (follower only). Emitted every 30 seconds.
ReceivingRequestRate	Number of requests per second this node is receiving (follower only). Emitted every 30 seconds.
SendingBandwidthRate	Number of bytes per second this node is sending (leader only). This value is undefined on single member clusters. Emitted every 30 seconds.
SendingRequestRate	Number of requests per second this node is sending (leader only). This value is undefined on single member clusters. Emitted every 30 seconds.
SentAppendRequests	Number of requests that this node has sent. Emitted every 30 seconds.
SetsFail	SetsFail operation count. Emitted every 30 seconds.
SetsSuccess	SetsSuccess operation count. Emitted every 30 seconds.
UpdateFail	UpdateFail operation count. Emitted every 30 seconds.
UpdateSuccess	UpdateSuccess operation count. Emitted every 30 seconds.
Watchers	Watchers operation count. Emitted every 30 seconds.

[Top](#)

Metron Agent

Default Origin Name: MetronAgent

Metric Name	Description
MessageAggregator.counterEventReceived	Lifetime number of CounterEvents aggregated in Metron.
MessageBuffer.droppedMessageCount	Lifetime number of intentionally dropped messages from Metron's batch writer buffer. Batch writing is performed over TCP/TLS only.
DopplerForwarder.sentMessages	Lifetime number of messages sent to Doppler regardless of protocol. Emitted every 30 seconds.
dropsondeAgentListener.currentBufferCount	Instantaneous number of Dropsonde messages read by UDP socket but not yet unmarshalled.
dropsondeAgentListener.receivedByteCount	Lifetime number of bytes of Dropsonde messages read by UDP socket.
dropsondeAgentListener.receivedMessageCount	Lifetime number of Dropsonde messages read by UDP socket.
dropsondeMarshaller.containerMetricMarshalled	Lifetime number of ContainerMetric messages marshalled.
dropsondeMarshaller.counterEventMarshalled	Lifetime number of CounterEvent messages marshalled.
dropsondeMarshaller.errorMarshalled	Lifetime number of Error messages marshalled.
dropsondeMarshaller.heartbeatMarshalled	Lifetime number of Heartbeat messages marshalled.
dropsondeMarshaller.httpStartStopMarshalled	Lifetime number of HttpStartStop messages marshalled.
dropsondeMarshaller.logMessageMarshalled	Lifetime number of LogMessage messages marshalled.
dropsondeMarshaller.marshalErrors	Lifetime number of errors when marshalling messages.
dropsondeMarshaller.valueMetricMarshalled	Lifetime number of ValueMetric messages marshalled.
dropsondeUnmarshaller.containerMetricReceived	Lifetime number of ContainerMetric messages unmarshalled.
dropsondeUnmarshaller.counterEventReceived	Lifetime number of CounterEvent messages unmarshalled.
dropsondeUnmarshaller.errorReceived	Lifetime number of Error messages unmarshalled.
dropsondeUnmarshaller.heartbeatReceived	DEPRECATED
dropsondeUnmarshaller.httpStartStopReceived	Lifetime number of HttpStartStop messages unmarshalled.
dropsondeUnmarshaller.logMessageTotal	Lifetime number of LogMessage messages unmarshalled.
dropsondeUnmarshaller.unmarshalErrors	Lifetime number of errors when unmarshalling messages.
dropsondeUnmarshaller.valueMetricReceived	Lifetime number of ValueMetric messages unmarshalled.
legacyAgentListener.currentBufferCount	Instantaneous number of Legacy messages read by UDP socket but not yet unmarshalled.
legacyAgentListener.receivedByteCount	Lifetime number of bytes of Legacy messages read by UDP socket.
legacyAgentListener.receivedMessageCount	Lifetime number of Legacy messages read by UDP socket.
memoryStats.lastGCPauseTimeNS	Duration of the last Garbage Collector pause in nanoseconds.
memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use.
memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use.
memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator.
memoryStats.numFrees	Lifetime number of memory deallocations.
memoryStats.numMallocs	Lifetime number of memory allocations.
numCpus	Number of CPUs on the machine.
numGoRoutines	Instantaneous number of active goroutines in the Doppler process.
tcp.sendErrorCount	Lifetime number of errors if writing to Doppler over TCP fails.
tcp.sentByteCount	Lifetime number of sent bytes to Doppler over TCP.
tcp.sentMessageCount	Lifetime number of sent messages to Doppler over TCP.
tls.sendErrorCount	Lifetime number of errors if writing to Doppler over TLS fails.
tls.sentByteCount	Lifetime number of sent bytes to Doppler over TLS. Emitted every 30 seconds.
tls.sentMessageCount	Lifetime number of sent messages to Doppler over TLS. Emitted every 30 seconds.
udp.sendErrorCount	Lifetime number of errors if writing to Doppler over UDP fails.
udp.sentByteCount	Lifetime number of sent bytes to Doppler over UDP.
udp.sentMessageCount	Lifetime number of sent messages to Doppler over UDP.

[Top](#)

Routing

Routing Release metrics have following origin names:

- [gorouter](#)
- [routing_api](#)
- [tcp_emitter](#)
- [tcp_router](#)

Default Origin Name: gorouter

Metric Name	Description
memoryStats.lastGCPauseTimeNS	Duration of the last Garbage Collector pause in nanoseconds. Emitted every 10 seconds.
memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use. Emitted every 10 seconds.
memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use. Emitted every 10 seconds.
memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator. Emitted every 10 seconds.
memoryStats.numFrees	Lifetime number of memory deallocations. Emitted every 10 seconds.
memoryStats.numMallocs	Lifetime number of memory allocations. Emitted every 10 seconds.
numCPUS	Number of CPUs on the machine. Emitted every 10 seconds.
numGoRoutines	Instantaneous number of active goroutines in the Doppler process. Emitted every 10 seconds.
logSenderTotalMessagesRead	Lifetime number of application log messages. Emitted every 5 seconds.
bad_gateways	Lifetime number of bad gateways. Emitted every 5 seconds.
latency	Time in milliseconds that the Gorouter took to handle requests to its application endpoints. Emitted per router request.
latency.{component}	Time in milliseconds that the Gorouter took to handle requests from each component to its endpoints. Emitted per router request.
registry_message.{component}	Lifetime number of route register messages received for each component. Emitted per route-register message.
unregistry_message.{component}	Lifetime number of route unregister messages received for each component. Emitted per route-unregister message.
rejected_requests	Lifetime number of bad requests received on Gorouter. Emitted every 5 seconds.
requests.{component}	Lifetime number of requests received for each component. Emitted every 5 seconds.
responses	Lifetime number of HTTP responses. Emitted every 5 seconds.
responses.2xx	Lifetime number of 2xx HTTP responses. Emitted every 5 seconds.
responses.3xx	Lifetime number of 3xx HTTP response. Emitted every 5 seconds.
responses.4xx	Lifetime number of 4xx HTTP response. Emitted every 5 seconds.
responses.5xx	Lifetime number of 5xx HTTP response. Emitted every 5 seconds.
responses.xxx	Lifetime number of other(non-(2xx-5xx)) HTTP response. Emitted every 5 seconds.
route_lookup_time	Time in nanoseconds to look up a request URL in the route table. Emitted per router request.
websocket_upgrades	Lifetime number of WebSocket upgrades. Emitted every 5 seconds.
websocket_failures	Lifetime number of WebSocket failures. Emitted every 5 seconds.
routed_app_requests	The collector sums up requests for all dea-{index} components for its output metrics. Emitted every 5 seconds.
total_requests	Lifetime number of requests received. Emitted every 5 seconds.
ms_since_last_registry_update	Time in millisecond since the last route register has been received. Emitted every 30 seconds.
total_routes	Current number of routes registered. Emitted every 30 seconds.
uptime	Uptime for router. Emitted every second.

Default Origin Name: routing_api

Metric Name	Description
memoryStats.lastGCPauseTimeNS	Duration of the last Garbage Collector pause in nanoseconds. Emitted every 10 seconds.

Metric Name	Description
memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use. Emitted every 10 seconds.
memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use. Emitted every 10 seconds.
memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator. Emitted every 10 seconds.
memoryStats.numFrees	Lifetime number of memory deallocations. Emitted every 10 seconds.
memoryStats.numMallocs	Lifetime number of memory allocations. Emitted every 10 seconds.
numCPUS	Number of CPUs on the machine. Emitted every 10 seconds.
numGoRoutines	Instantaneous number of active goroutines in the Doppler process. Emitted every 10 seconds.
key_refresh_events	Total number of events when fresh token was fetched from UAA. Emitted every 30 seconds.
total_http_routes	Number of HTTP routes in the routing table. Emitted every 30 seconds, or when there is a new HTTP route added. Interval for emitting this metric can be configured with manifest property <code>metrics_reporting_interval</code> .
total_http_subscriptions	Number of HTTP routes subscriptions. Emitted every 30 seconds. Interval for emitting this metric can be configured with manifest property <code>metrics_reporting_interval</code> .
total_tcp_routes	Number of TCP routes in the routing table. Emitted every 30 seconds, or when there is a new TCP route added. Interval for emitting this metric can be configured with manifest property <code>metrics_reporting_interval</code> .
total_tcp_subscriptions	Number of TCP routes subscriptions. Emitted every 30 seconds. Interval for emitting this metric can be configured with manifest property <code>metrics_reporting_interval</code> .
total_token_errors	Total number of UAA token errors. Emitted every 30 seconds. Interval for emitting this metric can be configured with manifest property <code>metrics_reporting_interval</code> .

Default Origin Name: tcp_emitter

Metric Name	Description
memoryStats.lastGCPauseTimeNS	Duration of the last Garbage Collector pause in nanoseconds. Emitted every 10 seconds.
memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use. Emitted every 10 seconds.
memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use. Emitted every 10 seconds.
memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator. Emitted every 10 seconds.
memoryStats.numFrees	Lifetime number of memory deallocations. Emitted every 10 seconds.
memoryStats.numMallocs	Lifetime number of memory allocations. Emitted every 10 seconds.
numCPUS	Number of CPUs on the machine. Emitted every 10 seconds.
numGoRoutines	Instantaneous number of active goroutines in the Doppler process. Emitted every 10 seconds.

Default Origin Name: router_configurer (bosch job tcp_router)

Metric Name	Description
memoryStats.lastGCPauseTimeNS	Duration of the last Garbage Collector pause in nanoseconds. Emitted every 10 seconds.
memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use. Emitted every 10 seconds.
memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use. Emitted every 10 seconds.
memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator. Emitted every 10 seconds.
memoryStats.numFrees	Lifetime number of memory deallocations. Emitted every 10 seconds.
memoryStats.numMallocs	Lifetime number of memory allocations. Emitted every 10 seconds.
numCPUS	Number of CPUs on the machine. Emitted every 10 seconds.
numGoRoutines	Instantaneous number of active goroutines in the Doppler process. Emitted every 10 seconds.
{session_id}.ConnectionTime	Average connection time to backend in current session. Emitted every 60 seconds per session ID. Interval value for this metric can be configured with manifest property <code>router_configurer.tcp_stats_collection_interval</code> .
{session_id}.CurrentSessions	Total number of current sessions. Emitted every 60 seconds per session ID. Interval value for this metric can be configured with manifest property <code>router_configurer.tcp_stats_collection_interval</code> .
AverageConnectTimeMs	Average backend response time (in ms). Emitted every 60 seconds. Interval value for this metric can be configured with manifest property <code>router_configurer.tcp_stats_collection_interval</code> .
	Average time spent in queue (in ms). Emitted every 60 seconds. Interval value for this metric can be

AvgQueueTimeMs	Metric Name	Description
TotalBackendConnectionErrors		Configured with manifest property <code>router_configurer.tcp_stats_collection_interval</code> .
TotalCurrentQueuedRequests		Total number of backend connection errors. Emitted every 60 seconds. Interval value for this metric can be configured with manifest property <code>router_configurer.tcp_stats_collection_interval</code> .

[Top](#)

Syslog Drain Binder

Default Origin Name: `syslog_drain_binder`

Metric Name	Description
<code>memoryStats.lastGCPauseTimeNS</code>	Duration of the last Garbage Collector pause in nanoseconds.
<code>memoryStats.numBytesAllocated</code>	Instantaneous count of bytes allocated and still in use.
<code>memoryStats.numBytesAllocatedHeap</code>	Instantaneous count of bytes allocated on the main heap and still in use.
<code>memoryStats.numBytesAllocatedStack</code>	Instantaneous count of bytes used by the stack allocator.
<code>memoryStats.numFrees</code>	Lifetime number of memory deallocations.
<code>memoryStats.numMallocs</code>	Lifetime number of memory allocations.
<code>numCPUS</code>	Number of CPUs on the machine.
<code>numGoRoutines</code>	Instantaneous number of active goroutines in the Doppler process.
<code>pollCount</code>	Number of times the syslog drain binder has polled the cloud controller for syslog drain bindings. Emitted every 30 seconds.
<code>totalDrains</code>	Number of syslog drains returned by cloud controller. Emitted every 30 seconds.

[Top](#)

Traffic Controller

Default Origin Name: `LoggregatorTrafficController`

Metric Name	Description
<code>dopplerProxy.containermetricsLatency</code>	Duration for serving container metrics via the containermetrics endpoint (milliseconds). Emitted every 30 seconds.
<code>dopplerProxy.recentlogsLatency</code>	Duration for serving recent logs via the recentLogs endpoint (milliseconds). Emitted every 30 seconds.
<code>memoryStats.lastGCPauseTimeNS</code>	Duration of the last Garbage Collector pause in nanoseconds.
<code>memoryStats.numBytesAllocated</code>	Instantaneous count of bytes allocated and still in use.
<code>memoryStats.numBytesAllocatedHeap</code>	Instantaneous count of bytes allocated on the main heap and still in use.
<code>memoryStats.numBytesAllocatedStack</code>	Instantaneous count of bytes used by the stack allocator.
<code>memoryStats.numFrees</code>	Lifetime number of memory deallocations.
<code>memoryStats.numMallocs</code>	Lifetime number of memory allocations.
<code>numCPUS</code>	Number of CPUs on the machine.
<code>numGoRoutines</code>	Instantaneous number of active goroutines in the Doppler process.
<code>Uptime</code>	Uptime for the Traffic Controller's process. Emitted every 30 seconds.
<code>LinuxFileDescriptor</code>	Number of file handles for the TrafficController's process.

[Top](#)

User Account and Authentication (UAA)

Default Origin Name: uaa

Metric Name	Description
audit_service.client_authentication_count	Number of successful client authentication attempts since the last startup. Emitted every 30 seconds.
audit_service.client_authentication_failure_count	Number of failed client authentication attempts since the last startup. Emitted every 30 seconds.
audit_service.principal_authentication_failure_count	Number of failed non-user authentication attempts since the last startup. Emitted every 30 seconds.
audit_service.principal_not_found_count	Number of times non-user was not found since the last startup. Emitted every 30 seconds.
audit_service.user_authentication_count	Number of successful authentications by the user since the last startup. Emitted every 30 seconds.
audit_service.user_authentication_failure_count	Number of failed user authentication attempts since the last startup. Emitted every 30 seconds.
audit_service.user_not_found_count	Number of times the user was not found since the last startup. Emitted every 30 seconds.
audit_service.user_password_changes	Number of successful password changes by the user since the last startup. Emitted every 30 seconds.
audit_service.user_password_failures	Number of failed password changes by the user since the last startup. Emitted every 30 seconds.

[Top](#)

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 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.

Prerequisites

- [BOSH CLI](#) installed locally.
- Spiff installed locally and added to your shell’s load path. 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 bandwidth, even when using the automated `provision_cf` script.

Step 1: Download Cloud Foundry BOSH Manifest

1. Run `bosh deployments` to identify the name of the current BOSH deployment:

```
$ bosh deployments
+-----+-----+
| Name | Release(s) | Stemcell(s)
+-----+-----+
| cf-example | cf-mysql/10 | bosh-vsphere-esxi-ubuntu-trusty-go_agent/2690.3 |
|          | cf/183.2   |           |
+-----+-----+
```

2. Run `bosh download manifest DEPLOYMENT-NAME LOCAL-SAVE-NAME` to download and save the current BOSH deployment manifest. You need this manifest to locate information about your databases. Replace `DEPLOYMENT-NAME` with the name of the current BOSH deployment. For this procedure, use `cf.yml` as the `LOCAL-SAVE-NAME`.

```
$ bosh download manifest cf-example cf.yml
Deployment manifest saved to `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: authorization_code,client_credentials,refresh_token  
  override: true  
  secret: example-nozzle  
  scope: openid,oauth.approvals,doppler.firehose  
  authorities: oauth.login,doppler.firehose
```

6. Save the deployment manifest file.

Step 3: Redeploy Cloud Foundry

1. Use the `bosh deployment` command to set the edited manifest file for your deployment.

```
$ bosh deployment cf.yml  
Deployment set to '/Users/example_user/workspace/bosh-lite(cf.yml'
```

2. Deploy your 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-incubator/example-nozzle'
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 status --uuid` to obtain your BOSH director UUID.

```
$ bosh status --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: Set Nozzle Deployment Manifest

Use the `bosh deployment` command to set the deployment manifest for the nozzle.

```
$ bosh deployment tmp/manifest_bosh_lite.yml  
Deployment set to '/Users/example_user/workspace/example-nozzle-release/templates/tmp/manifest_bosh_lite.yml'
```

Step 7: 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 create release --name example-nozzle  
Syncing blobs...  
...
```

Step 8: 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/qs1rjbmd1c5gb78m3_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 (b68dc9557ef296cb21c577c31ba97e2584a5154b)  
  
Jobs  
- example-nozzle (112e01c6ee91e8b268a42239e58e8e18e0360f58)  
  
License  
- none  
  
Uploading release
```

Step 9: 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 10: 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.

- Run `bosh ssh` to access the nozzle VM at the IP configured in the nozzle's manifest template `stu./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
```

- 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" counterEvent:
origin:"MetronAgent" eventType:CounterEvent timestamp:1443046218910360012 deployment:"cf-warden" job:"loggregator_trafficcontroller_z1" index:"0" ip:"203.0.113.146" counterEvent:
origin:"MetronAgent" eventType:CounterEvent timestamp:1443046218910252169 deployment:"cf-warden" job:"loggregator_trafficcontroller_z1" index:"0" ip:"203.0.113.146" counterEvent:
origin:"MetronAgent" eventType:CounterEvent timestamp:1443046218910294255 deployment:"cf-warden" job:"loggregator_trafficcontroller_z1" index:"0" ip:"203.0.113.146" counterEvent:
origin:"MetronAgent" eventType:CounterEvent timestamp:1443046218910318582 deployment:"cf-warden" job:"loggregator_trafficcontroller_z1" index:"0" ip:"203.0.113.146" counterEvent:
origin:"MetronAgent" eventType:CounterEvent timestamp:1443046218910339088 deployment:"cf-warden" job:"loggregator_trafficcontroller_z1" index:"0" ip:"203.0.113.146" counterEvent:
origin:"MetronAgent" eventType:CounterEvent timestamp:1443046218910379199 deployment:"cf-warden" job:"loggregator_trafficcontroller_z1" index:"0" ip:"203.0.113.146" counterEvent:
origin:"MetronAgent" eventType:CounterEvent timestamp:1443046218910394886 deployment:"cf-warden" job:"loggregator_trafficcontroller_z1" index:"0" ip:"203.0.113.146" counterEvent:
origin:"router_0" eventType:HttpStartStop timestamp:1443046219105062148 deployment:"cf-warden" job:"router_z1" index:"0" ip:"203.0.113.22" httpStartStop: peerType:Client method:
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 method:
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:
```

Cloud Foundry Data Sources

Page last updated:

Currently, Cloud Foundry logs and metrics come from several sources:

- Loggregator is the next generation logging and metrics system for Cloud Foundry. It aggregates metrics from applications and CF system components and streams these out to the CF cli or to third party log management services.
- The BOSH Health Monitor continually listens for one ‘heartbeat’ per minute from each deployed 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.
- Logs from CF components can also be forwarded directly to your own server, bypassing loggregator. See [Loggregator for Operators](#) for more information.

Currently, 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 Elastic Runtime.

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

- Run `cf add-plugin-repo REPO_NAME URL` to add the Cloud Foundry Community plugin repository to your cf CLI plugins.

```
$ cf add-plugin-repo CF-Community https://plugins.cloudfoundry.org
```

- Run `cf install-plugin PLUGIN-NAME -r PLUGIN-REPO` to install the Firehose plugin from the CF Community plugin repository.

```
$ cf install-plugin "Firehose Plugin" -r CF-Community
```

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
```

Pivotal Cloud Foundry Release Notes

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

PCF Release Notes

- [Pivotal Elastic Runtime Release Notes](#)
 - [Pivotal Operations Manager Release Notes](#)
 - [PCF Isolation Segment Release Notes](#)
 - [Stemcell Release Notes](#)
-

PCF Services Release Notes

- [App Distribution for PCF](#) [x]
- [DataStax Enterprise for PCF](#) [x]
- [GemFire for PCF](#) [x]
- [MySQL for PCF](#) [x]
- [PCF Application Watchdog](#) [x]
- [PCF JMX Bridge](#) [x]
- [PCF Log Search](#) [x]
- [PCF Metrics](#) [x]
- [PCF Service Broker for AWS](#) [x]
- [Pivotal Tracker for PCF](#) [x]
- [Push Notification Service for PCF](#) [x]
- [Rabbit MQ for PCF](#) [x]
- [Redis for PCF](#) [x]
- [Riak CS for PCF](#) [x]
- [Session State Caching Powered by GemFire](#) [x]
- [Single Sign-On for PCF](#) [x]
- [Spring Cloud Services on PCF](#) [x]

Pivotal Elastic Runtime v1.10 Release Notes

Component Versions

Versions 1.10.0 and higher versions of Elastic Runtime consist of the following component versions:

Component	Version
Stemcell	3363.10
binary-buildpack	1.0.9
capi	1.21.0*
cf	252*
cf-autoscaling	84
cf-mysql	34
cf-networking	0.16.0*
cflinuxfs2-rootfs	1.50.0
consul	152
diego	1.7.1*
dotnet-core-buildpack	1.0.11
etcd	93
garden-runc	1.1.1
go-buildpack	1.7.18
java-offline-buildpack	3.13
loggregator	77
mysql-backup	1.32.0
mysql-monitoring	7.2.0
nats	15
nfs-volume	0.1.5
nodejs-buildpack	1.5.29
notifications	35
notifications-ui	27
php-buildpack	4.3.26
pivotal-account	1.4.1
postgres	13
push-apps-manager-release	660.7.2
python-buildpack	1.5.15
routing	0.146.0*
ruby-buildpack	1.6.34
service-backup	18.0.3
staticfile-buildpack	1.3.17
uaa	27

* 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 Cloud Foundry Elastic Runtime v1.10 is documented in the [Upgrading Pivotal Cloud Foundry](#) topic.

When upgrading to v1.10, be aware of the following upgrade considerations:

- You must upgrade first to a version of Elastic Runtime v1.9.x in order to successfully upgrade to v1.10.
- If you are currently using any of the following services in your PCF deployment, then you must upgrade and configure the tiles before upgrading to PCF v1.10:
 - **RabbitMQ for PCF.** Upgrade to RabbitMQ for PCF v1.7.13 or later, and deselect the **Use non-secure communication for metrics** checkbox. For more information about RabbitMQ for PCF, see the [RabbitMQ for PCF documentation](#).
 - **Redis for PCF.** Upgrade to Redis for PCF v1.7.3 or later, and deselect the **Use non-secure communication for metrics** checkbox. For more information about Redis for PCF, see the [Redis for PCF documentation](#).
- Some partner service tiles may be incompatible with PCF v1.10. Pivotal is working with partners to ensure their tiles are being updated to work with the latest versions of PCF.

For information about which partner service releases are currently compatible with PCF v1.10, review the appropriate partners services release documentation at <http://docs.pivotall.io>, or contact the partner organization that produces the tile.

About Advanced Features

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

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

New Features in Elastic Runtime v1.10.0

This section describes new features of the release.

Container-to-Container Networking

The Elastic Runtime tile offers a Container-to-Container Networking feature that puts applications in their own overlay network.

Container-to-Container Networking is available as an advanced feature in Elastic Runtime.

For more information, see the [Container-to-Container Networking](#) topic.

Volume Services

This release provides general support for volume services inside of application containers.

Additionally, the Elastic Runtime ships with an NFS Volume Service Broker as an advanced feature.

For more information, see [Using an External File System \(Volume Services\)](#) and [Enabling NFS Volume Services](#).

Secure etcd Cluster

This release removes the etcd Proxy VM, ensuring all communication to the etcd cluster happens over a secure connection.

Cloud Foundry API Rate Limiting

The CF API can enforce rate limits for users and clients.

Limits can be set for authenticated and unauthenticated clients and expire over a rolling hour-long window.

You can enable these API rate limits in the **Advanced Features** section of the Elastic Runtime tile.

For more information, see the *Deploying Elastic Runtime* topic for the IaaS where you are deploying PCF. For example, if you are deploying PCF on Google Cloud Platform (GCP), see the [Deploying Elastic Runtime on GCP](#) topic.

Cloud Foundry Diego Operator Toolkit

All Diego VMs now include an operator toolkit, known as `cfdot`, for interacting with your Diego components.

For more details about `cfdot`, see the [cfdot documentation](#).

Multi-node Cloud Controller Clock

The Cloud Controller Clock has been outfitted to allow multiple instances of the VM to run in parallel.

Operators can scale the instance count for the VM to fit their needs. For example, operators might want to change the instance count to 2 or 3 so they have a clock in each availability zone.

For more information, see [High Availability in Cloud Foundry](#).

Router Performance Improvements

The Routers can now be configured to maintain a number of keep-alive connections.

Reusing connections allows for HTTP performance improvements as the underlying connection does not need to be re-established on every request.

For more information, see the [Router Idle Keepalive Connections](#) and the [Deploying Elastic Runtime](#) topic for the IaaS where you are deploying PCF. For example, if you are deploying PCF on Google Cloud Platform (GCP), see the [Deploying Elastic Runtime on GCP](#) topic.

Disable Default SSH Access for New Applications

Previously, application SSH access was previously enabled globally as a feature in Cloud Foundry.

In addition to the global setting, operators can now choose to disable SSH access for new applications.

Choosing to disable SSH access for new applications requires that developers enable SSH access on a per-application basis.

For more information, see [Configuring SSH Access for PCF](#).

Azure Blobstore Support

The Elastic Runtime now supports using Azure Storage as a backend for the platform file storage.

For more information, see the [Deploying Elastic Runtime on Azure](#) topic.

Improvements Internal MySQL Diagnostics and Availability

The internal MySQL database cluster now includes healthcheck thresholds.

These thresholds can be configured to match your MySQL load balancer thresholds so that failover is seamless.

For information on internal MySQL load balancer configuration, see the [Deploying Elastic Runtime](#) topic for the IaaS where you are deploying PCF. For example, if you are deploying PCF on Google Cloud Platform (GCP), see the [Deploying Elastic Runtime on GCP](#) topic.

Additionally, the MySQL Monitor job now includes a tool called `mysql-diag` that provides some diagnostic information about your MySQL cluster.

For more information on the `mysql-diag` tool, see [Diagnosing problems with Elastic Runtime MySQL or the Pivotal MySQL Tile](#).

Global Container Max Inflight Configuration

Diego now provides a configuration option for limiting the number of containers allowed to be in a “starting” state at any one time.

By default, the setting limits the number of containers in the “starting” state to 200.

This setting prevents Diego from scheduling too much work for your platform to handle, preventing a possible cascading failure.

This configuration is available as the **Max Inflight Container Starts** on the **Application Containers** screen in Elastic Runtime.

To configure this feature, see [Setting a Maximum Number of Started Containers](#).

For more information on preventing platform overload during upgrade, see also [Upgrade Considerations for Selecting File Storage in Pivotal Cloud Foundry](#) and [Managing Diego Cell Limits During Upgrade](#) topics.

gRPC in Loggregator

The Loggregator system now uses the [gRPC](#) protocol for secure and reliable 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.

Since Loggregator now uses the gRPC protocol, your deployment may see an increase in Loggregator message throughput.

For more information on scaling Dopplers, see the [Upgrading Pivotal Cloud Foundry](#) and [Loggregator Guide for Cloud Foundry Operators](#) topics.

Known Issues

This section lists known issues for PCF Elastic Runtime.

cf logs Connection Issue

When tailing logs using the `cf logs` or `cf logs --recent` command, the cf CLI reports a connection issue. Users may encounter errors similar to the following:

```
Warning: error tailing logs
Error dialing loggregator server: websocket: bad handshake.
Please ask your Cloud Foundry Operator to check the platform configuration
(loggregator endpoint is wss://loggregator.example.com:443).
```

or

```
FAILED
Error dialing loggregator server: unexpected EOF
Please ask your Cloud Foundry Operator to check the platform configuration
(loggregator endpoint is wss://loggregator.example.com:443).
```

Solution: Upgrade to cf CLI version 6.23 or later. After you upgrade, if you still encounter the connection issue, make sure you log out and log in again using `cf logout` and `cf login`.

Messages Received Firehose Metric Invalid in JMX Bridge

In PCF v1.10, the recommended metric for monitoring firehose message throughput is changing from `DopplerServer.listeners.receivedMessageCount` to `DopplerServer.listeners.receivedEnvelopes`. However, the latter metric utilizes a new tagged metric structure that the JMX Bridge does not handle. If you consume metrics with JMX Bridge, you should continue to use `DopplerServer.listeners.receivedMessageCount` for monitoring firehose message throughput.

- As of ERT 1.10.0, `DopplerServer.listeners.receivedMessageCount` is not an accurate metric for all possible firehose traffic. This is being patched. JMX Bridge will require this patch version.
- As of ERT 1.10.1, users of JMX Bridge can reliably monitor throughput using `DopplerServer.listeners.receivedMessageCount`.

Missing JMX Bridge Metrics

For users of the JMX Bridge, tagged metrics (such as the throughput metric) are not output by the new Loggregator.

Pivotal Cloud Foundry Ops Manager v1.10 Release Notes

How to Upgrade

The procedure for upgrading to Pivotal Cloud Foundry (PCF) Ops Manager v1.10 is documented in the [Upgrading Pivotal Cloud Foundry](#) topic.

1.10.0

Version 1.10.0 of Ops Manager consists of the following component versions:

Versions
BOSH Director: v261.4
bosh-init : v0.0.100
Stemcell: 3363.x
AWS CPI: TK
Azure CPI: TK
Google Cloud Platform CPI: TK
OpenStack CPI: TK
vSphere CPI: TK

New Features in Ops Manager v1.10.0

Ops Manager API

Operators can now use the API to automate more tasks, such as configuring tiles, uploading tiles, and fetching upgrades from Pivotal Network. To view the Ops Manager API documentation, browse to <https://YOUR-OPS-MANAGER-FQDN/docs>.

New Endpoints

The Ops Manager API adds the following endpoints:

- `PUT /api/v0/staged/director/network_and_az`: Operators can assign a network and singleton availability zone for the Ops Manager Director tile.
- `GET /api/v0/staged/cloud_config`: Operators can fetch a BOSH cloud config based on the staged state of Ops Manager.
- `GET /api/v0/deployed/cloud_config`: Operators can fetch a BOSH cloud config based on the deployed state of Ops Manager.
- `GET /api/v0/certificateAuthorities`: Operators can list all root certificate authorities for Ops Manager.
- `POST /api/v0/certificateAuthorities`: Operators can create a root certificate authority.
- `POST /api/v0/certificateAuthorities/active/regenerate`: Operators can rotate non-configurable certificates by deleting all non-configuration certificates and then regenerating them.

 **Note:** For more information about using the Ops Manager API for certificate rotation, see the [Certificate Rotation](#) section below.

- `POST /api/v0/certificateAuthorities/generate`: Operators can generate an additional root certificate authority.
- `POST /api/v0/certificateAuthorities/:certificate_authority_guid/activate`: Operators can activate a root certificate authority, and make all others inactive.
- `DELETE /api/v0/certificateAuthorities/:certificate_authority_guid`: Operators can delete a specific inactive certificate authority from Ops Manager.
- `POST /api/v0/staged/installations/commit`: Operators can save the installation state as if the deployment was triggered by clicking **Apply**

Changes, preparing the installation manifest. Operators can then fetch the manifest using the

```
/api/v0/deployed/product/[product_id]/manifest
```

endpoint.

Certificate Rotation

Ops Manager ships with a Certificate Authority (CA). This CA generates certificates that are used for communication across various PCF components. These certificates are non-configurable.

Ops Manager now allows you to regenerate non-configurable TLS/SSL certs using the API. See the [Rotating Non-Configurable TLS/SSL Certificates](#) topic for more information.

Ops Manager also allows operators to use an API endpoint to add their own custom CA authority. If added, this custom CA will be used to sign all non-configurable certificates and deployed across all relevant PCF components.

SHA-2

Previous versions of PCF used SHA-1 hashes, while PCF v1.10 uses SHA-2 by default. Both Operations Manager installations and certificates provided by PCF use SHA-2 hashes.

You cannot obtain SHA-2 by upgrading existing environments to PCF v1.10. However, you can convert existing SHA-1 hashes into SHA-2 hashes by rotating your Ops Manager certificates by following the procedure in [Regenerating and Rotating Non-Configurable TLS/SSL Certificates](#).

Azure Government Cloud

Operators can now deploy Ops Manager in an Azure Government Cloud environment. For more information, see the [Deploying PCF on Azure Government Cloud](#) topic.

Smart Errands

Errands are scripts that Ops Manager runs to automate tasks. In PCF 1.10, Ops Manager provides three configurations for errands:

- **On:** The errand always runs.
- **Off:** The errand never runs.
- **When Changed:** The errand runs only if the associated product manifest has changed since the last successful errand run.

If a tile developer does not specify a default configuration for an errand, then Ops Manager applies the **When Changed** configuration.

Operators should set errand configurations to **On** for any tile that pushes apps. For example, if your deployment includes the [Single Sign-On](#), [Push Notification Service](#), or [PCF Metrics](#) tiles, the errands for these tiles must run automatically to keep up with buildpack patches.

See the [Managing Errands in Ops Manager](#) topic for more information.

BOSH Updates

The following list sections updates in the new BOSH version that are not exposed by Ops Manager, but may be helpful for operators and tile developers for improving their workflows.

BOSH Director

Ops Manager v1.10.0 uses BOSH v261, which includes the following changes to the BOSH Director:

- **Context IDs for Tasks:** The [Tasks](#) endpoint for the BOSH Director API includes a `context_id` property. This feature is particularly useful in tracking multiple BOSH tasks that are related. For example, PCF On-Demand Services uses it to track the multiple BOSH tasks involved in the `cf create-service` process.
- **HM Alerts as BOSH Events:** You can now view BOSH Health Monitor alerts, such as process restarts and monit alerts, with the

`bosh events` command. The `bosh events` command provides a historic view of your system, including events such as the creation of VMs. For more information, see the [BOSH Events documentation](#).

- **Event filtering query parameters:** If you use v2 of the BOSH CLI, you can now use filtering flags for detailed BOSH event recording and querying. For more information, see the [BOSH Events documentation](#).

For a full list of updates and fixes in the new BOSH version that Ops Manager uses, see the [BOSH release notes](#), beginning with v261.

Stemcell

Ops Manager v1.10.0 uses Stemcell 3363.10. Here are some of the major changes in Stemcell 3363:

- Additional auditd rules
- A new group, `bosh_sshers`, assigned to the `vcap` user. Users must belong to this group to use SSH.
- Log Agent API access events are sent in CEF format to syslog via the `vcap.agent` topic.

For a full list of updates and fixes in the new stemcell that Ops Manager uses, see the [stemcell 3363 release notes](#).

PCF Isolation Segment v1.10 Release Notes

Component Versions

Versions 1.10.0 and higher versions of PCF Isolation Segment consist of the following component versions:

Component	Version
Stemcell	3312.12
cflinuxfs2-rootfs	1.50.0
consul	152
diego	1.7.1
garden-runc	1.1.1
loggregator	77
routing	0.145.0

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

About PCF Isolation Segment

PCF Isolation Segment is a new tile available for installation with PCF v.1.10.

[Isolation segments](#) provide dedicated pools of resources where you can deploy apps and isolate workloads. Using isolation segments separates app resources as completely as if they were in different CF deployments but avoids redundant management and network complexity.

For more information on using isolation segments in your deployment, see the [Managing Isolation Segments](#) topic.

How to Install

The procedure for installing PCF Isolation Segment v1.10 is documented in the [Installing PCF Isolation Segment](#) topic.

To install a PCF Isolation Segment, you must first install PCF v1.10.

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 1.10.x.

3363.10

Release Date: March 8, 2017

- Bumps Ubuntu stemcells for USN-3220-2: Linux kernel (Xenial HWE) vulnerability

3363.9

Release Date: February 22, 2017

Changes

- Bumps Ubuntu stemcells for USN-3208-2: Linux kernel (Xenial HWE) vulnerabilities
- Fixes excessive “out of memory” errors in kernel
 - <https://bugs.launchpad.net/ubuntu/+source/linux/+bug/1655842>
- Fixes regression to rsyslog by locking it down again to rsyslog 8.22.0

Agent:

- Fixes Azure stemcell persistent disk formatting
- Fixes Warden stemcells SSH access

3363.1

Release Date: February 15, 2017

Reported Problems:

- DO NOT USE azure stemcell as it may cause data loss.
- rsyslog version updated to 8.24.0, regressing on issue #1537
- Out of memory errors still exists in Kernel 4.4.0.62
 - will be fixed around Feb 20.

Changes:

- Fixes double -hvm- suffix problem for AWS Light stemcells

3363

Release Date: February 15, 2017

Reported Problems

- DO NOT USE azure stemcell as it may cause data loss.
- Out of memory errors still exists in Kernel 4.4.0.62
 - will be fixed around Feb 20.
- rsyslog version updated to 8.24.0, regressing on issue #1537
- AWS Light stemcell has incorrect name once imported
- BOSH SSH does not work on BOSH Lite

Changes

- Add more auditd rules
- Fix CentOS initramfs to load necessary kernel modules
- Disable boot loader login
- Increasing tcp_max_sync_backlog
- Disabling any DSA host keys
- Add bosh_sshers group and assign it to vcap user
 - Only allow users in bosh_sshers group to SSH

Agent

- Log Agent API access events in CEF format to syslog (vcap.agent topic)
- Allow configuring swap size through env.bosh.swap_size (example: env.bosh.swap_size: 0)
- Prepare for SHA2 releases
- Allow setting fetching to work with base64 encoded user data
- Do not delaycompress in logrotate