Pivotal

PRODUCT DOCUMENTATION

Pivotal Container Service (PKS)

Version 1.3

Published: 17 July 2019

Table of Contents

Pivotal Container Service (PKS)

Page last updated:

Pivotal Container Service (PKS) enables operators to provision, operate, and manage enterprise-grade Kubernetes clusters using BOSH and Pivotal Ops Manager.

Overview

PKS uses the On-Demand Broker to deploy Cloud Foundry Container Runtime, a BOSH release that offers a uniform way to instantiate, deploy, and manage highly available Kubernetes clusters on a cloud platform using BOSH.

After operators install the PKS tile on the Ops Manager Installation Dashboard, developers can provision Kubernetes clusters using the PKS Command Line Interface (PKS CLI), and run container-based workloads on the clusters with the Kubernetes CLI, kubectl.

PKS is available as part of Pivotal Cloud Foundry or as a stand-alone product.

What PKS Adds to Kubernetes

The following table details the features that PKS adds to the Kubernetes platform.

Feature	Included in K8s	Included in PKS
Single tenant ingress	✓	✓
Secure multi-tenant ingress		✓
Stateful sets of pods	✓	✓
Multi-container pods	✓	✓
Rolling upgrades to pods	✓	✓
Rolling upgrades to cluster infrastructure		✓
Pod scaling and high availability	✓	✓
Cluster provisioning and scaling		✓
Monitoring and recovery of cluster VMs and processes		✓
Persistent disks	✓	✓
Secure container registry		✓
Embedded, hardened operating system		✓

Features

PKS has the following features:

- **Kubernetes compatibility**: Constant compatibility with current stable release of Kubernetes
- **Production-ready**: Highly available from applications to infrastructure, with no single points of failure
- **BOSH advantages**: Built-in health checks, scaling, auto-healing and rolling upgrades
- **Fully automated operations**: Fully automated deploy, scale, patch, and upgrade experience
- **Multi-cloud**: Consistent operational experience across multiple clouds
- **GCP APIs access**: The Google Cloud Platform (GCP) Service Broker gives applications access to the Google Cloud APIs, and Google Container Engine (GKE) consistency enables the transfer of workloads from or to GCP

On vSphere, PKS supports deploying and running Kubernetes clusters in air-gapped environments.

PKS Components

The PKS control plane contains the following components:

- An On-Demand Broker that deploys Cloud Foundry Container Runtime (CFCR), an open-source project that provides a solution for deploying and managing Kubernetes clusters using BOSH.
- A Service Adapter
- The PKS API

For more information about the PKS control plane, see PKS Cluster Management.

For a detailed list of components and supported versions by a particular PKS release, see the PKS Release Notes.

PKS Concepts

For conceptual information about PKS, see PKS Concepts.

PKS Prerequisites

For information about the resource requirements for installing PKS, see the topic that corresponds to your cloud provider:

- vSphere Prerequisites and Resource Requirements
- vSphere with NSX-T Version Requirements and Hardware Requirements for PKS on vSphere with NSX-T
- GCP Prerequisites and Resource Requirements
- AWS Prerequisites and Resource Requirements
- Azure Prerequisites and Resource Requirements

Preparing to Install PKS

To install PKS, you must deploy one of the following versions of Ops Manager:

- Ops Manager v2.4.3 and earlier in the v2.4 version line
- Ops Manager v2.3.9 and earlier in the v2.3 version line

You use Ops Manager to install and configure PKS.

If you are installing PKS to vSphere, you can also configure integration with NSX-T and Harbor.

Consult the following table for compatibility information:

IaaS	Ops Manager v2.3.1+ or v2.4.x	NSX-T	Harbor
vSphere	Required	Available	Available
GCP	Required	Not Available	Available
AWS	Required	Not Available	Available
Azure	Ops Manager v2.4.0 through v2.4.3 or v2.3.0 through v2.3.9 is required for installing PKS on Azure.	Not Available	Available

For more information about compatibility and component versions, see the PKS Release Notes.

For information about preparing your environment before installing PKS, see the topic that corresponds to your cloud provider:

- vSphere
- vSphere with NSX-T Integration
- GCP
- AWS
- Azure

Installing PKS

For information about installing PKS, see *Installing PKS* for your IaaS:

- vSphere
- vSphere with NSX-T Integration
- Google Cloud Platform (GCP)
- Amazon Web Services (AWS)
- Microsoft Azure (Azure)

Upgrading PKS

For information about upgrading the PKS tile and PKS-deployed Kubernetes clusters, see Upgrading PKS Overview.

Managing PKS

For information about configuring authentication, creating users, and managing your PKS deployment, see Managing PKS.

Using PKS

For information about using the PKS CLI to create and manage Kubernetes clusters, see Using PKS.

Backing Up and Restoring PKS

For information about using BOSH Backup and Restore (BBR) to back up and restore PKS, see Backing Up and Restoring PKS.

PKS Security

For information about security in PKS, see PKS Security.

Diagnosing and Troubleshooting PKS

For information about diagnosing and troubleshooting issues installing or using PKS, see Diagnosing and Troubleshooting PKS.

Please send any feedback you have to pks-feedback@pivotal.io.

PKS Release Notes

Page last updated:

This topic contains release notes for Pivotal Container Service (PKS) v1.3.x.

v1.3.6

Release Date: April 8, 2019

Product Snapshot

Element	Details
Version	v1.3.6
Release date	April 8, 2019
Compatible Ops Manager versions	v2.3.1+, v2.4.0+
Stemcell version	v170.15
Kubernetes version	v1.12.7
On-Demand Broker version	v0.24
CFCR	v0.25.11
NSX-T versions *	v2.3.1, v2.4.0.1
NCP version	v2.4.0
Docker version	v18.06.3-ce CFCR v0.25.11

Note: Ops Manager v2.3.10 and later in the v2.3 version line and Ops Manager v2.4.4 and later in the v2.4 version line do not support PKS v1.3 on Azure. Before deploying PKS v1.3 on Azure, you must install Ops Manager v2.3.9 or earlier in the 2.3 version line or Ops Manager v2.4.3 or earlier in the 2.4 version line.

Note: NSX-T v2.4 implements a new Policy API that PKS v1.3.6 does not support. If you are using NSX-T v2.4 with PKS 1.3.6, you must use the “Advanced Networking” tab in NSX Manager to create, read, update, and delete network object required for PKS.

vSphere Version Requirements

If you are installing PKS on vSphere or vSphere with NSX−T, note that Ops Manager and PKS support the following vSphere component versions:

Versions	Editions
• VMware vSphere 6.7 U1 EP06 (ESXi670-201901001) – for NSX-T 2.4 • VMware vSphere 6.7 U1 • VMware vSphere 6.7.0 • VMware vSphere 6.5 U2 P03 (ESXi650-201811002) – for NSX-T 2.4 • VMware vSphere 6.5 U2 • VMware vSphere 6.5 U1	• vSphere Enterprise Plus • vSphere with Operations Management Enterprise Plus

Note: VMware vSphere 6.7 is only supported with Ops Manager v2.3.1 or later and NSX−T v2.3.

For more information, see Upgrading vSphere in an NSX Environment in the VMware documentation.

Feature Support by IaaS

	AWS	Azure	GCP	vSphere	vSphere with NSX-T
Automatic Kubernetes Cluster API load balancer					✓
HTTP proxy				✓	✓
Multi-AZ storage				✓	✓
Per-namespace subnets					✓
Service `type:LoadBalancer`	✓*	✓	✓		✓

Upgrade Path

The supported upgrade paths to PKS v1.3.6 are as follows:

- PKS v1.3.4 or later

When upgrading to NSX-T 2.4:

- Use the official VMware NSX-T Data Center 2.4 build.
- Apply the NSX-T v2.4.0.1 hot-patch. For more information, see VMware Knowledge Base KB article 67499.
- To obtain the NSX-T v2.4.0.1 hot-patch, open a support ticket with VMware Global Support Services (GSS) for NSX-T Engineering.

Features

New features and changes in this release:

- Telemetry property `environment_provider`.
- Support for `nsx-cf-cni` with 2.4.0.12511604.
- Remaining plans to the osb-proxy configuration.

Breaking Changes and Known Issues

Breaking Change: Heapster is deprecated in PKS v1.3.x, and Kubernetes has retired Heapster. For more information, see the kubernetes-retired/heapster repository on GitHub.

PKS v1.3.6 has the following known issues:

Azure Resource Group Field in the Kubernetes Cloud Provider Is Ignored

The PKS tile's **Resource Group** configuration is ignored on Azure IaaS platform deployments. On Azure, the PKS VM is always deployed to the same **Resource Group** as the Ops Manager and BOSH VMs. The **Resource Group** field is in the PKS tile's **Kubernetes Cloud Provider** section.

NSX-T Upgrades from v2.3.X to v2.4.0.1 Fail for Bare Metal Edge Node

Upgrading NSX-T v2.3.X to v2.4.0.1 fails for Bare Metal Edge Nodes.

If you are using a Bare Metal Edge Nodes, please refrain from upgrading NSX-T v2.3.x to NSX-T v2.4.0.1.

Worker Nodes with Small Ephemeral Disks Can Cause Upgrade Failure

PKS deploys packages to the ephemeral disk, `/var/vcap/data`, during installations and upgrades. If master or worker node VMs have ephemeral disks smaller than 8 GB, the disk can fill during an upgrade and cause the upgrade to fail. Cluster upgrades can present error messages such as the following:

```
{"time":999999999,"error":{"code":450001,"message":"Response exceeded maximum allowed length"}}
```

Workaround: In the plans you use to deploy clusters, ensure that worker and master node ephemeral disks are set to greater than 8 GB. For plan configuration instructions, see the Plans section of the *Installing PKS* topic for your IaaS.

This issue should not affect new installations of PKS v1.3.x as the default ephemeral disk size in plans is larger than 8 GB.

PKS Flannel Network Gets Out of Sync with Docker Bridge Network (cni0)

When VMs have been powered down for multiple days, turning them back on and issuing a `bosh recreate` to re-create the VMs causes the pods to get stuck in a `ContainerCreating` state.

Workaround: See PKS Flannel network gets out of sync with docker bridge network (cni0) in the Pivotal Knowledge Base.

Cluster Upgrades from PKS v1.3.0 on Azure Fail If Services Are Exposed

If you install PKS v1.3.0 on Azure, clusters might fail with the following error when you upgrade to PKS v1.3.1 or later:

```
result: 1 of 2 post-start scripts failed. Failed Jobs: kubelet. Successful Jobs: bosh-dns
```

This issue is caused by a timeout condition. The issue affects nodes hosting Kubernetes pods that are exposed externally by a Kubernetes service.

New cluster creations and cluster scaling operations are not affected by this issue.

Workaround: If you install PKS on Azure and experience this issue, contact Support for assistance.

The kubelet customization feature is only enabled for Plan 1

PKS 1.3.4 introduces the ability to configure kubelet startup parameters `system-reserved` and `evction-hard` within a plan. This capability is only functional in Plan 1 for PKS 1.3.4 and will be enabled in additional plans in the next release.

v1.3.5

Release Date: March 28, 2019

Product Snapshot

Element	Details
Version	v1.3.5
Release date	March 28, 2019
Compatible Ops Manager versions	v2.3.1+, v2.4.0+
Stemcell version	v170.15
Kubernetes version	v1.12.7
On-Demand Broker version	v0.24
CFCR	v0.25.11
NSX-T versions *	v2.2, v2.3.0.2, v2.3.1
NCP version	v2.3.2
Docker version	v18.06.3-ce CFCR v0.25.11

Note: Ops Manager v2.3.10 and later in the v2.3 version line and Ops Manager v2.4.4 and later in the v2.4 version line do not support PKS v1.3 on Azure. Before deploying PKS v1.3 on Azure, you must install Ops Manager v2.3.9 or earlier in the 2.3 version line or Ops Manager v2.4.3 or earlier in the 2.4 version line.

vSphere Version Requirements

If you are installing PKS on vSphere or vSphere with NSX-T, note that Ops Manager and PKS support the following vSphere component versions:

Versions	Editions
• VMware vSphere 6.7 U1 EP06 (ESXi670-201901001) – for NSX-T 2.4 • VMware vSphere 6.7 U1 • VMware vSphere 6.7.0 • VMware vSphere 6.5 U2 P03 (ESXi650-201811002) – for NSX-T 2.4 • VMware vSphere 6.5 U2 • VMware vSphere 6.5 U1	• vSphere Enterprise Plus • vSphere with Operations Management Enterprise Plus

Note: VMware vSphere 6.7 is only supported with Ops Manager v2.3.1 or later and NSX-T v2.3.

For more information, see Upgrading vSphere in an NSX Environment in the VMware documentation.

Feature Support by IaaS

	AWS	Azure	GCP	vSphere	vSphere with NSX-T
Automatic Kubernetes Cluster API load balancer					✓
HTTP proxy				✓	✓
Multi-AZ storage				✓	✓
Per-namespace subnets					✓
Service `type:LoadBalancer`	✓*	✓	✓		✓

Upgrade Path

The supported upgrade paths to PKS v1.3.5 are as follows:

- PKS v1.3.4 or later

Features

New features and changes in this release:

- Support for Kubernetes v1.12.7.
- **Fix**: CVE-2019-1002101. Kubernetes v1.12.7 address this CVE.
- **Fix**: CVE-2019-9946. Kubernetes v1.12.7 address this CVE.

Breaking Changes and Known Issues

Breaking Change: Heapster is deprecated in PKS v1.3.x, and Kubernetes has retired Heapster. For more information, see the kubernetes-retired/heapster repository on GitHub.

PKS v1.3.5 has the following known issues:

Worker Nodes with Small Ephemeral Disks Can Cause Upgrade Failure

PKS deploys packages to the ephemeral disk, `/var/vcap/data`, during installations and upgrades. If master or worker node VMs have ephemeral disks smaller than 8 GB, the disk can fill during an upgrade and cause the upgrade to fail. Cluster upgrades can present error messages such as the following:

```
{"time":999999999,"error":{"code":450001,"message":"Response exceeded maximum allowed length"}}
```

Workaround: In the plans you use to deploy clusters, ensure that worker and master node ephemeral disks are set to greater than 8 GB. For plan configuration instructions, see the Plans section of the *Installing PKS* topic for your IaaS.

This issue should not affect new installations of PKS v1.3.x as the default ephemeral disk size in plans is larger than 8 GB.

PKS Flannel Network Gets Out of Sync with Docker Bridge Network (cni0)

When VMs have been powered down for multiple days, turning them back on and issuing a `bosh recreate` to re-create the VMs causes the pods to get stuck in a `ContainerCreating` state.

Workaround: See PKS Flannel network gets out of sync with docker bridge network (cni0) in the Pivotal Knowledge Base.

Cluster Upgrades from PKS v1.3.0 on Azure Fail If Services Are Exposed

If you install PKS v1.3.0 on Azure, clusters might fail with the following error when you upgrade to PKS v1.3.1 or later:

```
result: 1 of 2 post-start scripts failed. Failed Jobs: kubelet. Successful Jobs: bosh-dns
```

This issue is caused by a timeout condition. The issue affects nodes hosting Kubernetes pods that are exposed externally by a Kubernetes service.

New cluster creations and cluster scaling operations are not affected by this issue.

Workaround: If you install PKS on Azure and experience this issue, contact Support for assistance.

The kubelet customization feature is only enabled for Plan 1

PKS 1.3.4 introduces the ability to configure kubelet startup parameters `system-reserved` and `evction-hard` within a plan. This capability is only functional in Plan 1 for PKS 1.3.4 and will be enabled in additional plans in the next release.

---

v1.3.4

Release Date: March 26, 2019

Product Snapshot

Element	Details
Version	v1.3.4
Release date	March 26, 2019
Compatible Ops Manager versions	v2.3.1+, v2.4.0+
Stemcell version	v170.15
Kubernetes version	v1.12.6
On-Demand Broker version	v0.24
CFCR	v0.25.11
NSX-T versions *	v2.2, v2.3.0.2, v2.3.1
NCP version	v2.3.2
Docker version	v18.06.3-ce CFCR v0.25.11

Note: Ops Manager v2.3.10 and later in the v2.3 version line and Ops Manager v2.4.4 and later in the v2.4 version line do not support PKS v1.3 on Azure. Before deploying PKS v1.3 on Azure, you must install Ops Manager v2.3.9 or earlier in the 2.3 version line or Ops Manager v2.4.3 or earlier in the 2.4 version line.

vSphere Version Requirements

If you are installing PKS on vSphere or vSphere with NSX−T, note that Ops Manager and PKS support the following vSphere component versions:

Versions	Editions
• VMware vSphere 6.7 U1 EP06 (ESXi670-201901001) – for NSX-T 2.4 • VMware vSphere 6.7 U1 • VMware vSphere 6.7.0 • VMware vSphere 6.5 U2 P03 (ESXi650-201811002) – for NSX-T 2.4 • VMware vSphere 6.5 U2 • VMware vSphere 6.5 U1	• vSphere Enterprise Plus • vSphere with Operations Management Enterprise Plus

Note: VMware vSphere 6.7 is only supported with Ops Manager v2.3.1 or later and NSX−T v2.3.

For more information, see Upgrading vSphere in an NSX Environment in the VMware documentation.

Feature Support by IaaS

	AWS	Azure	GCP	vSphere	vSphere with NSX-T
Automatic Kubernetes Cluster API load balancer					✓
HTTP proxy				✓	✓
Multi-AZ storage				✓	✓
Per-namespace subnets					✓
Service `type:LoadBalancer`	✓*	✓	✓		✓

Upgrade Path

The supported upgrade paths to PKS v1.3.3 are as follows:

- **When upgrading from PKS v1.3.x**: PKS v1.3.1 or later
- **When upgrading from PKS v1.2.x**: PKS v1.2.8 or later

Features

New features and changes in this release:

- Custom DNS configuration for Kubernetes clusters using NSX-T and Network Profiles. For more information, see DNS Configuration for Kubernetes Clusters in *Defining Network Profiles*.
- Support for NSX-T NCP v2.3.2. For more information, see the VMware NSX Container Plug-in 2.3.2 Release Notes.
- Support for additional plans. Operators can configure up to ten sets of resource types, or **Plans**, in the PKS tile. All plans except the first can made available or unavailable to developers deploying clusters. **Plan 1** must be configured and made available as a default for developers.
- Kubelet customization. You can enable Kubelet to reserve compute resources for system daemons by configuring the startup parameters `system-reserved` and `eviction-hard` in the **Plans** pane of the PKS tile. For more information, see the *Plans* section of the *Installing PKS* topic for your IaaS, such as Installing PKS on vSphere.
- **Fix**: CVE-2019-1002100. Kubernetes v1.12.6 address this CVE.
- **Fix**: Updated the Telemetry URL.

- **Fix**: Resolved an issue where vSphere Cloud Provider configuration could fail if credentials contained non-alphanumeric characters. For example, `#`, `\`, and `"`.

Breaking Changes and Known Issues

Breaking Change: Heapster is deprecated in PKS v1.3.x, and Kubernetes has retired Heapster. For more information, see the kubernetes-retired/heapster repository on GitHub.

PKS v1.3.4 has the following known issues:

Master and Worker Nodes with Small Ephemeral Disks Can Cause Upgrade Failure

PKS deploys packages to the ephemeral disk, `/var/vcap/data`, during installations and upgrades. If master and worker node VMs have ephemeral disks smaller than 8 GB, the disk can fill during an upgrade and cause the upgrade to fail. Cluster upgrades can present error messages such as the following:

```
{"time":999999999,"error":{"code":450001,"message":"Response exceeded maximum allowed length"}}
```

Workaround: In the plans you use to deploy clusters, ensure that the master and worker node ephemeral disks are set to greater than 8 GB. For plan configuration instructions, see the *Plans* section of the *Installing PKS* topic for your IaaS, such as Installing PKS on vSphere.

This issue should not affect new installations of PKS v1.3.x as the default ephemeral disk size in plans is larger than 8 GB.

PKS Flannel Network Gets Out of Sync with Docker Bridge Network (cni0)

When VMs have been powered down for multiple days, turning them back on and issuing a `bosh recreate` to re-create the VMs causes the pods to get stuck in a `ContainerCreating` state.

Workaround: See PKS Flannel network gets out of sync with docker bridge network (cni0) in the Pivotal Knowledge Base.

Cluster Upgrades from PKS v1.3.0 on Azure Fail If Services Are Exposed

If you install PKS v1.3.0 on Azure, clusters might fail with the following error when you upgrade to PKS v1.3.1 or later:

```
result: 1 of 2 post-start scripts failed. Failed Jobs: kubelet. Successful Jobs: bosh-dns
```

This issue is caused by a timeout condition. The issue affects nodes hosting Kubernetes pods that are exposed externally by a Kubernetes service.

New cluster creations and cluster scaling operations are not affected by this issue.

Workaround: If you install PKS on Azure and experience this issue, contact Support for assistance.

Kubelet Customization Feature Only Enabled for Plan 1

PKS v1.3.4 introduces the ability to configure Kubelet startup parameters `system-reserved` and `eviction-hard` within a plan. For more information, see the *Plans* section of the *Installing PKS* topic for your IaaS, such as Installing PKS on vSphere.

This feature is only functional in Plan 1 for PKS v1.3.4 and will be enabled in additional plans in the next release.

v1.3.3

Release Date: February 22, 2019

Product Snapshot

Element	Details
Version	v1.3.3
Release date	February 22, 2019
Compatible Ops Manager versions	v2.3.1+, v2.4.0+
Stemcell version	v170.15
Kubernetes version	v1.12.5
On-Demand Broker version	v0.24
CFCR	v0.25.9
NSX-T versions *	v2.2, v2.3.0.2, v2.3.1
NCP version	v2.3.1
Docker version	v18.06.3-ce CFCR v0.25.9

Note: Ops Manager v2.3.10 and later in the v2.3 version line and Ops Manager v2.4.4 and later in the v2.4 version line do not support PKS v1.3 on Azure. Before deploying PKS v1.3 on Azure, you must install Ops Manager v2.3.9 or earlier in the 2.3 version line or Ops Manager v2.4.3 or earlier in the 2.4 version line.

Feature Support by IaaS

	AWS	Azure	GCP	vSphere	vSphere with NSX-T
Automatic Kubernetes Cluster API load balancer					✓
HTTP proxy				✓	✓
Multi-AZ storage				✓	✓
Per-namespace subnets					✓
Service `type:LoadBalancer`	✓*	✓	✓		✓

Upgrade Path

The supported upgrade paths to PKS v1.3.3 are as follows:

- **When upgrading from PKS v1.3.x**: PKS v1.3.1 or v1.3.2
- **When upgrading from PKS v1.2.x**: PKS v1.2.8 through v1.2.11

Features

New features and changes in this release:

- **Fix**: CVE-2019-5736. This release updates the version of Docker deployed by PKS to v18.06.3-ce. This Docker version addresses a runc vulnerability whereby a malicious image could run in privileged mode and elevate to root access on worker nodes. Docker v18.06.2-ce, deployed by PKS v1.3.2, did not contain the correct compiled binary. This Docker version includes the correct runc binary to address the CVE.

Breaking Changes and Known Issues

Breaking Change: Heapster is deprecated in PKS v1.3.x, and Kubernetes has retired Heapster. For more information, see the kubernetes-retired/heapster repository on GitHub.

PKS v1.3.3 has the following known issues:

PKS Flannel Network Gets Out of Sync with Docker Bridge Network (cni0)

When VMs have been powered down for multiple days, turning them back on and issuing a `bosh recreate` to re-create the VMs causes the pods to get stuck in a `ContainerCreating` state.

Workaround: See PKS Flannel network gets out of sync with docker bridge network (cni0) in the Pivotal Knowledge Base.

Deploy Fails if vSphere Master Credentials Field Has Special Characters Without Quotes

If you install PKS on vSphere and you enter credentials in the **vCenter Master Credentials** field of the **Kubernetes Cloud Provider** pane of the PKS tile that contain special characters, such as `#`, `$`, `,`, `!`, or `-`, your deployment might fail with the following error:

```
ServerFaultCode: Cannot complete login due to an incorrect user name or password.
```

Workaround: If you install PKS on vSphere **without** NSX-T integration, place quotes around the credentials in the cloud provider configuration. For example, `"SomeP4$$w0rd#!"`. Then redeploy the PKS tile by clicking **Apply Changes**.

If you install PKS on vSphere **with** NSX-T integration, avoid using special characters in this field until this issue is resolved.

Cluster Upgrades from PKS v1.3.0 on Azure Fail If Services Are Exposed

If you install PKS v1.3.0 on Azure, clusters might fail with the following error when you upgrade to PKS v1.3.1 or later:

```
result: 1 of 2 post-start scripts failed. Failed Jobs: kubelet. Successful Jobs: bosh-dns
```

This issue is caused by a timeout condition. The issue affects nodes hosting Kubernetes pods that are exposed externally by a Kubernetes service.

New cluster creations and cluster scaling operations are not affected by this issue.

Workaround: If you install PKS on Azure and experience this issue, contact Support for assistance.

v1.3.2

Release Date: February 13, 2019

Product Snapshot

Element	Details
Version	v1.3.2
Release date	February 13, 2019
Compatible Ops Manager versions	v2.3.1+, v2.4.0+
Stemcell version	v170.15
Kubernetes version	v1.12.4
On-Demand Broker version	v0.24
CFCR	v0.25.8
NSX-T versions *	v2.2, v2.3.0.2, v2.3.1
NCP version	v2.3.1
Docker version	v18.06.2-ce CFCR v0.25.8

Note: Ops Manager v2.3.10 and later in the v2.3 version line and Ops Manager v2.4.4 and later in the v2.4 version line do not support PKS v1.3 on Azure. Before deploying PKS v1.3 on Azure, you must install Ops Manager v2.3.9 or earlier in the 2.3 version line or Ops Manager v2.4.3 or earlier in the 2.4 version line.

Feature Support by IaaS

	AWS	Azure	GCP	vSphere	vSphere with NSX-T
Automatic Kubernetes Cluster API load balancer					✓
HTTP proxy				✓	✓
Multi-AZ storage				✓	✓
Per-namespace subnets					✓
Service `type:LoadBalancer`	✓*	✓	✓		✓

Upgrade Path

The supported upgrade paths to PKS v1.3.2 are as follows:

- **When upgrading from PKS v1.3.x**: PKS v1.3.1
- **When upgrading from PKS v1.2.x**: PKS v1.2.8 or v1.2.9

Features

New features and changes in this release:

- **Fix**: CVE-2019-3779. This fix addresses a vulnerability where certs signed by the Kubernetes API could be used to gain access to a PKS-deployed cluster's etcd service.
- **Fix**: CVE-2019-3780. This fixes a regression bug in PKS where vCenter IaaS credentials intended for the vSphere Cloud Provider were written on worker node VM disks.
- **Fix**: Clusters can now be successfully created if there are pre-existing Kubernetes clusters using the same hostname.

Breaking Changes and Known Issues

Breaking Change: Heapster is deprecated in PKS v1.3.x, and Kubernetes has retired Heapster. For more information, see the kubernetes-retired/heapster repository on GitHub.

PKS v1.3.2 has the following known issues:

PKS Flannel Network Gets Out of Sync with Docker Bridge Network (cni0)

When VMs have been powered down for multiple days, turning them back on and issuing a `bosh recreate` to re-create the VMs causes the pods to get stuck in a `ContainerCreating` state.

Workaround: See PKS Flannel network gets out of sync with docker bridge network (cni0) in the Pivotal Knowledge Base.

Deploy Fails if vSphere Master Credentials Field Has Special Characters Without Quotes

If you install PKS on vSphere and you enter credentials in the **vCenter Master Credentials** field of the **Kubernetes Cloud Provider** pane of the PKS tile that contain special characters, such as `#`, `$`, `,`, `!`, or `-`, your deployment might fail with the following error:

```
ServerFaultCode: Cannot complete login due to an incorrect user name or password.
```

Workaround: If you install PKS on vSphere **without** NSX-T integration, place quotes around the credentials in the cloud provider configuration. For example, `"SomeP4$$w0rd#!"`. Then redeploy the PKS tile by clicking **Apply Changes**.

If you install PKS on vSphere **with** NSX-T integration, avoid using special characters in this field until this issue is resolved.

Cluster Upgrades from PKS v1.3.0 on Azure Fail If Services Are Exposed

If you install PKS v1.3.0 on Azure, clusters might fail with the following error when you upgrade to either PKS v1.3.1 or later:

```
result: 1 of 2 post-start scripts failed. Failed Jobs: kubelet. Successful Jobs: bosh-dns
```

This issue is caused by a timeout condition. The issue affects nodes hosting Kubernetes pods that are exposed externally by a Kubernetes service.

New cluster creations and cluster scaling operations are not affected by this issue.

Workaround: If you install PKS on Azure and experience this issue, contact Support for assistance.

v1.3.1

Release Date: February 8, 2019

WARNING: PKS v1.3.1 and earlier includes a critical CVE. Follow the procedures in the PKS upgrade approach for CRITICAL CVE article in the Pivotal Support Knowledge Base to perform an upgrade to PKS v1.3.2.

Product Snapshot

Element	Details
Version	v1.3.1
Release date	February 8, 2019
Compatible Ops Manager versions	v2.3.1+, v2.4.0+
Stemcell version	v170.15
Kubernetes version	v1.12.4
On-Demand Broker version	v0.24
CFCR	v0.25.8
NSX-T versions *	v2.2, v2.3.0.2, v2.3.1
NCP version	v2.3.1
Docker version	v18.06.1-ce CFCR v0.25.8

Note: Ops Manager v2.3.10 and later in the v2.3 version line and Ops Manager v2.4.4 and later in the v2.4 version line do not support PKS v1.3 on Azure. Before deploying PKS v1.3 on Azure, you must install Ops Manager v2.3.9 or earlier in the 2.3 version line or Ops Manager v2.4.3 or earlier in the 2.4 version line.

vSphere Version Requirements

If installing PKS on vSphere or vSphere with NSX−T, please note Ops Manager and PKS support the following vSphere component versions:

Versions	Editions
• VMware vSphere 6.7 U1 EP06 (ESXi670-201901001) – for NSX-T 2.4 • VMware vSphere 6.7 U1 • VMware vSphere 6.7.0 • VMware vSphere 6.5 U2 P03 (ESXi650-201811002) – for NSX-T 2.4 • VMware vSphere 6.5 U2 • VMware vSphere 6.5 U1	• vSphere Enterprise Plus • vSphere with Operations Management Enterprise Plus

Note: VMware vSphere 6.7 is only supported with Ops Manager v2.3.1 or later and NSX−T v2.3.

For more information, see Upgrading vSphere in an NSX Environment in the VMware documentation.

Feature Support by IaaS

	AWS	Azure	GCP	vSphere	vSphere with NSX-T
Automatic Kubernetes Cluster API load balancer					✓
HTTP proxy				✓	✓
Multi-AZ storage				✓	✓
Per-namespace subnets					✓
Service `type:LoadBalancer`	✓*	✓	✓		✓

Upgrade Path

The supported upgrade paths to PKS v1.3.1 are as follows:

- **When upgrading from PKS v1.3.x**: PKS v1.3.0
- **When upgrading from PKS v1.2.x**: PKS v1.2.7 or v1.2.8

Follow the procedures in the PKS upgrade approach for CRITICAL CVE article in the Pivotal Support Knowledge Base to perform an upgrade to PKS v1.3.2.

Features

New features and changes in this release:

- Certificates for the Etcd instance for each Kubernetes cluster provisioned by PKS are generated with a four-year lifetime and signed by a new Etcd Certificate Authority (CA).
- **Fix**: Upgrading PKS no longer fails during upgrades if there are Kubernetes clusters with duplicate hostnames.
- **Fix**: Deploying PKS no longer fails if an entry in the **No Proxy** field contains special characters such as (`-`) character.
- **Fix**: The Kubernetes API now responds with the CA certificate that signed the Kubernetes cluster's certificate so that customer scripts such as the get-pks-k8s-config.sh tool will function again.

Breaking Changes and Known Issues

Breaking Change: Heapster is deprecated in PKS v1.3.x, and Kubernetes has retired Heapster. For more information, see the kubernetes-retired/heapster repository on GitHub.

PKS v1.3.1 has the following known issues:

PKS Flannel Network Gets out of Sync with Docker Bridge Network (cni0)

When VMs have been powered down for multiple days, turning them back on and issuing a `bosh recreate` to re-create the VMs causes the pods to get stuck in a `ContainerCreating` state.

Workaround: See PKS Flannel network gets out of sync with docker bridge network (cni0) in the Pivotal Knowledge Base.

Deploy Fails if vSphere Master Credentials Field Has Special Characters Without Quotes

If you install PKS on vSphere and you enter credentials in the **vCenter Master Credentials** field of the **Kubernetes Cloud Provider** pane of the PKS tile that contain special characters, such as `#`, `$`, `,`, `!`, or `-`, your deployment might fail with the following error:

```
ServerFaultCode: Cannot complete login due to an incorrect user name or password.
```

Workaround: If you install PKS on vSphere **without** NSX-T integration, place quotes around the credentials in the cloud provider configuration. For example, `"SomeP4$$w0rd#!"`. Then redeploy the PKS tile by clicking **Apply Changes**.

If you install PKS on vSphere **with** NSX-T integration, avoid using special characters in this field until this issue is resolved.

Cluster Upgrades from PKS v1.3.0 on Azure Fail If Services Are Exposed

If you install PKS v1.3.0 on Azure, clusters might fail with the following error when you upgrade to either PKS v1.3.1 or later:

```
result: 1 of 2 post-start scripts failed. Failed Jobs: kubelet. Successful Jobs: bosh-dns
```

This issue is caused by a timeout condition. The issue affects nodes hosting Kubernetes pods that are exposed externally by a Kubernetes service.

New cluster creations and cluster scaling operations are not affected by this issue.

v1.3.0 - Withdrawn

Release Date: January 16, 2019

This release has been removed from Pivotal Network because it has a known vulnerability. This issue has been fixed in PKS v1.3.1.

Product Snapshot

Element	Details
Version	v1.3.0
Release date	January 16, 2019
Compatible Ops Manager versions	v2.3.1+, v2.4.0+
Stemcell version	v170.15
Kubernetes version	v1.12.4
On-Demand Broker version	v0.24
NSX-T versions *	v2.2, v2.3.0.2, v2.3.1
NCP version	v2.3.1
Docker version	v18.06.1-ce CFCR

* PKS v1.3 supports NSX-T v2.2 and v2.3 with the following caveats:

- To use the network profile features available in PKS v1.3, you must use NSX-T v2.3.
- NSX-T 2.3.0 has a known critical issue: ESX hosts lose network connectivity rendering the host inaccessible from network (60293). This issue is patched in NSX-T v2.3.0.2 and fixed in the NSX-T v2.3.1 release.

Note: Ops Manager v2.3.10 and later in the v2.3 version line and Ops Manager v2.4.4 and later in the v2.4 version line do not support PKS v1.3 on Azure. Before deploying PKS v1.3 on Azure, you must install Ops Manager v2.3.9 or earlier in the 2.3 version line or Ops Manager v2.4.3 or earlier in the 2.4 version line.

vSphere Version Requirements

If installing PKS on vSphere or vSphere with NSX−T, please note Ops Manager and PKS support the following vSphere component versions:

Versions	Editions
• VMware vSphere 6.7 U1 EP06 (ESXi670-201901001) – for NSX-T 2.4	

• VMware vSphere 6.7 U1 • VMware vSphere 6.7.0 • VMware vSphere 6.5 U2 P03 (ESXi650-201811002) – for NSX-T 2.4 • VMware vSphere 6.5 U2 • VMware vSphere 6.5 U1	• vSphere Enterprise Plus • vSphere with Operations Management Enterprise Plus

Note: VMware vSphere 6.7 is only supported with Ops Manager v2.3.1 or later and NSX−T v2.3.

For more information, see Upgrading vSphere in an NSX Environment in the VMware documentation.

Feature Support by IaaS

	AWS	Azure	GCP	vSphere	vSphere with NSX-T
Automatic Kubernetes Cluster API load balancer					✓
HTTP proxy				✓	✓
Multi-AZ storage				✓	✓
Per-namespace subnets					✓
Service `type:LoadBalancer`	✓*	✓	✓		✓

* For more information about configuring Service `type:LoadBalancer` on AWS, see the Access Workloads Using an Internal AWS Load Balancer section of *Deploying and Exposing Basic Workloads*.

Upgrade Path

The supported upgrade paths to PKS v1.3.0 are from PKS v1.2.5 and later.

For more information, see Upgrading PKS and Upgrading PKS with NSX-T.

Note: Upgrading from PKS v1.2.5+ to PKS v1.3.x causes all certificates to be automatically regenerated. The old certificate authority is still trusted, and has a validity of one year. But the new certificates are signed with a new certificate authority, which is valid for four years.

Features

New features and changes in this release:

- Support for PKS on Azure. For more information, see Azure.
- BOSH Backup and Restore (BBR) for single-master clusters. For more information, see Back Up Cluster Deployments in *Backing Up PKS*, and Restore PKS Clusters in *Restoring PKS*.
- Routable pods on NSX-T. For more information, see Routable Pod Networks in *Defining Network Profiles*.
- Large size NSX-T load balancers with Bare Metal NSX-T edge nodes. For more information, see Hardware Requirements for PKS on vSphere with NSX-T.
- HTTP proxy for NSX-T components. For more information, see Using Proxies with PKS on NSX-T.
- Ability to specify the size of the Pods IP Block subnet using a network profile. For more information, see Pod Subnet Prefix in *Defining Network Profiles*.
- Support for bootstrap security groups, custom floating IPs, and edge router selection using network profiles. For more information, see Bootstrap Security Group, Custom Floating IP Pool, and Edge Router Selection in *Defining Network Profiles*.
- Support for sink resources in air-gapped environments.
- Support for creating sink resources with the PKS Command Line Interface (PKS CLI). For more information, see Creating Sink Resources.
- Sink resources include both pod logs as well as events from the Kubernetes API. These events are combined in a shared format that provides operators with a robust set of filtering and monitoring options. For more information, see Monitoring PKS with Sinks.
- Support for multiple NSX-T Tier-0 (T0) logical routers for use with PKS multi-tenant environments. For more information, see Configuring Multiple Tier-0 Routers for Tenant Isolation.
- Support for multiple PKS foundations on the same NSX-T. For more information, see Implementing a Multi-Foundation PKS Deployment.
- Smoke tests errand that uses the PKS CLI to create a Kubernetes cluster and then delete it. If the creation or deletion fails, the errand fails and the

installation of the PKS tile is aborted. For more information, see the *Errands* section of the *Installing PKS* topic for your IaaS, such as Installing PKS on vSphere.

- Support for scaling down the number of worker nodes. For more information, see Scaling Existing Clusters.
- Support for defining the CIDR range for Kubernetes pods and services on Flannel networks. For more information, see the*Networking* section of the *Installing PKS* topic for your IaaS, such as Installing PKS on vSphere.
- Kubernetes v1.12.4.
- **Bug Fix**: The **No Proxy** property for vSphere now accepts wildcard domains like `*.example.com` and `example.com`. See Networking in *Installing PKS on vSphere* for more information.
- **Bug Fix**: The issue with NSX-T where special characters in username and password doesn't work is resolved.
- **Security Fix**: CVE 2018-18264: This CVE allows unauthenticated secret access to the Kubernetes Dashboard.
- **Security Fix**: CVE-2018-15759: This CVE contains an insecure method of verifying credentials. A remote unauthenticated malicious user may make many requests to the service broker with a series of different credentials, allowing them to infer valid credentials and gain access to perform broker operations.

Breaking Changes and Known Issues

Breaking Change: Heapster is deprecated in PKS v1.3, and Kubernetes has retired Heapster. For more information, see the kubernetes-retired/heapster repository on GitHub.

PKS v1.3.0 has the following known issues:

Upgrades Fail When Clusters Share an External Hostname

If you use the same external hostname across more than one PKS-deployed Kubernetes cluster, upgrades from PKS v1.2.x to PKS v1.3.0 might fail. The external hostname is the value you set with either the `-e` or `--external-hostname` argument when you created the cluster. For more information, see Create a Kubernetes Cluster.

PKS v1.3.0 introduces restrictions that prevent you from deploying clusters with duplicate hostnames, so this issue does not affect upgrades from PKS v1.3.0 and later.

If you have existing clusters that use the same external hostname, do not upgrade to PKS v1.3.x. Contact your Support representative for more information.

Upgrades Fail with a Hyphen in the No Proxy Field on vSphere

If you install PKS on vSphere and you enable the **HTTP/HTTPS Proxy** setting, you cannot use the `-` character in the **No Proxy** field. Entering `-` in the **No Proxy** field can cause validation errors when trying to upgrade to PKS v1.3.0. For more information, see the Networking section of *Installing PKS on vSphere*.

If you experience this issue during an upgrade, contact Support for a hotfix that will be applied in a future PKS v1.3.x release.

PKS Flannel Network Gets Out of Sync with Docker Bridge Network (cni0)

When VMs have been powered down for multiple days, turning them back on and issuing a `bosh recreate` to re-create the VMs causes the pods to get stuck in a `ContainerCreating` state.

Workaround: See PKS Flannel network gets out of sync with docker bridge network (cni0) in the Pivotal Knowledge Base.

Deploy Fails if vSphere Master Credentials Field Has Special Characters Without Quotes

If you install PKS on vSphere and you enter credentials in the **vCenter Master Credentials** field of the **Kubernetes Cloud Provider** pane of the PKS tile that contain special characters, such as `#`, `$`, `,`, `!`, or `-`, your deployment might fail with the following error:

```
ServerFaultCode: Cannot complete login due to an incorrect user name or password.
```

Workaround: If you install PKS on vSphere **without** NSX-T integration, place quotes around the credentials in the cloud provider configuration. For example, `"SomeP4$$w0rd#!"`. Then redeploy the PKS tile by clicking **Apply Changes**.

If you install PKS on vSphere **with** NSX-T integration, avoid using special characters in this field until this issue is resolved.

PKS Selects the First AZ Only During Cluster Creation

If the first availability zone (AZ) used by a plan with multiple AZs runs out of resources, cluster creation fails with an error like the following:

```
L Error: CPI error 'Bosh::Clouds::CloudError' with message 'No valid placement found for requested memory: 4096
```

Explanation: BOSH creates VMs for your PKS deployment using a round-robin algorithm. It tries to create the first VM in the first AZ that your plan uses. If the first AZ runs out of resources, cluster creation fails and BOSH does not try to create more VMs.

Please send any feedback you have to pks-feedback@pivotal.io.

PKS Concepts

Page last updated:

This topic describes Pivotal Container Service (PKS) concepts. See the following sections:

- PKS Cluster Management
- PKS API Authentication
- Load Balancers in PKS
- VM Sizing for PKS Clusters
- PKS Telemetry
- PAS and PKS Deployments with Ops Manager
- Sink Architecture in PKS

Please send any feedback you have to pks-feedback@pivotal.io.

PKS Cluster Management

This topic describes how Pivotal Container Service (PKS) manages the deployment of Kubernetes clusters.

Overview

Users interact with PKS and PKS-deployed Kubernetes clusters in two ways:

- Deploying Kubernetes clusters with BOSH and managing their lifecycle. These tasks are performed using the PKS Command Line Interface (PKS CLI) and the PKS control plane.
- Deploying and managing container-based workloads on Kubernetes clusters. These tasks are performed using the Kubernetes CLI, `kubectl`.

Cluster Lifecycle Management

The PKS control plane enables users to deploy and manage Kubernetes clusters.

For communicating with the PKS control plane, PKS provides a command line interface, the PKS CLI. See Installing the PKS CLI for installation instructions.

PKS Control Plane Overview

The PKS control plane manages the lifecycle of Kubernetes clusters deployed using PKS. The control plane allows users to do the following through the PKS CLI:

- View cluster plans
- Create clusters
- View information about clusters
- Obtain credentials to deploy workloads to clusters
- Scale clusters
- Delete clusters
- Create and manage network profiles for VMware NSX-T

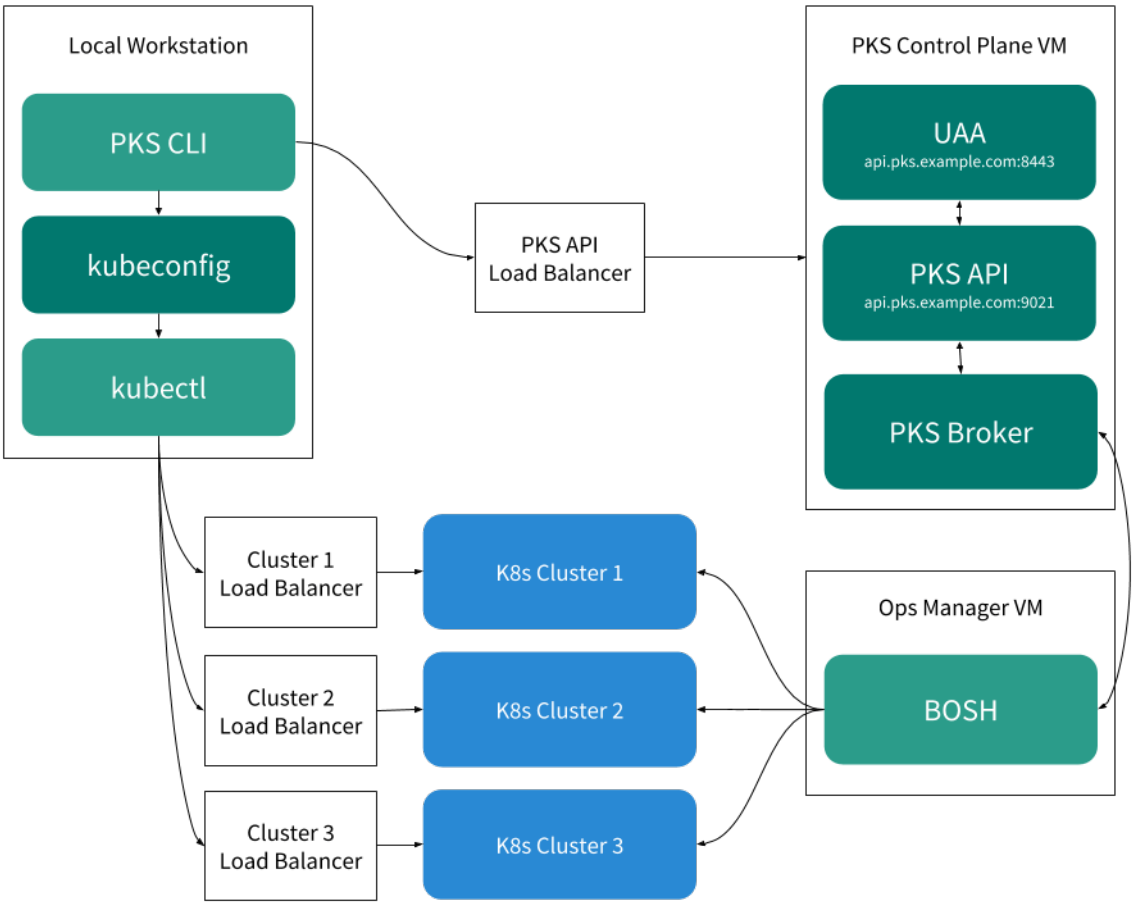
In addition, the PKS control plane can upgrade all existing clusters using the **Upgrade all clusters** BOSH errand. For more information, see Upgrade Kubernetes Clusters in *Upgrading PKS*.

PKS Control Plane Architecture

The PKS control plane is deployed on a single VM that includes the following components:

- The PKS API server
- The PKS Broker
- A User Account and Authentication (UAA) server

The following illustration shows how these components interact:



The PKS API Load Balancer is used for AWS, GCP, and vSphere without NSX-T deployments. If PKS is deployed on vSphere with NSX-T, a DNAT rule is configured for the PKS API host so that it is accessible. For more information, see the Share the PKS API Endpoint section in *Installing PKS on vSphere with NSX-T Integration*.

UAA

When a user logs in to or logs out of the PKS API through the PKS CLI, the PKS CLI communicates with UAA to authenticate them. The PKS API permits only authenticated users to manage Kubernetes clusters. For more information about authenticating, see PKS API Authentication.

UAA must be configured with the appropriate users and user permissions. For more information, see Managing Users in PKS with UAA.

PKS API

Through the PKS CLI, users instruct the PKS API server to deploy, scale up, and delete Kubernetes clusters as well as show cluster details and plans. The PKS API can also write Kubernetes cluster credentials to a local kubeconfig file, which enables users to connect to a cluster through `kubectl`.

The PKS API sends all cluster management requests, except read-only requests, to the PKS Broker.

PKS Broker

When the PKS API receives a request to modify a Kubernetes cluster, it instructs the PKS Broker to make the requested change.

The PKS Broker consists of an On-Demand Service Broker and a Service Adapter. The PKS Broker generates a BOSH manifest and instructs the BOSH Director to deploy or delete the Kubernetes cluster.

For PKS deployments on vSphere with NSX-T, there is an additional component, the PKS NSX-T Proxy Broker. The PKS API communicates with the PKS NSX-T Proxy Broker, which in turn communicates with the NSX Manager to provision the Node Networking resources. The PKS NSX-T Proxy Broker then forwards the request to the On-Demand Service Broker to deploy the cluster.

Cluster Workload Management

PKS users manage their container-based workloads on Kubernetes clusters through `kubectl`. For more information about `kubectl`, see Overview of kubectl in the Kubernetes documentation.

Please send any feedback you have to pks-feedback@pivotal.io.

PKS API Authentication

Page last updated:

This topic describes how the Pivotal Container Service (PKS) API works with User Account and Authentication (UAA) to manage authentication and authorization in your PKS deployment.

Authenticating PKS API Requests

Before users can log in and use the PKS CLI, you must configure PKS API access with UAA. For more information, see Configuring PKS API Access with UAA.

You use the UAA Command Line Interface (UAAC) to target the UAA server and request an access token for the UAA admin user. If your request is successful, the UAA server returns the access token. The UAA admin access token authorizes you to make requests to the PKS API using the PKS CLI and grant cluster access to new or existing users. For more information, see Grant Cluster Access in *Managing Users in PKS with UAA*.

When a user with cluster access logs in to the PKS CLI, the CLI requests an access token for the user from the UAA server. If the request is successful, the UAA server returns an access token to the PKS CLI. When the user runs PKS CLI commands, for example, `pks clusters`, the CLI sends the request to the PKS API server and includes the user's UAA token.

The PKS API sends a request to the UAA server to validate the user's token. If the UAA server confirms that the token is valid, the PKS API uses the cluster information from the PKS broker to respond to the request. For example, if the user runs `pks clusters`, the CLI returns a list of the clusters that the user is authorized to manage.

Routing to the PKS API Control Plane VM

The PKS API server and the UAA server use different port numbers on the control plane VM. For example, if your PKS API domain is `api.pks.example.com`, you can reach your PKS API and UAA servers at the following URLs:

Server	URL
PKS API	api.pks.example.com:9021
UAA	api.pks.example.com:8443

Refer to **Ops Manager > Pivotal Container Service > PKS API > API Hostname (FQDN)** for your PKS API domain.

Load balancer implementations differ by deployment environment. For PKS deployments on GCP, AWS, or vSphere without NSX-T, you configure a load balancer to access the PKS API when you install the PKS tile. For more information, see the Configure External Load Balancer section of *Installing PKS* for your IaaS.

For procedures that describe routing to the PKS control plane VM, see the Configure External Load Balancer section of *Installing PKS* for your IaaS.

For overview information about load balancers in PKS, see Load Balancers in PKS Deployments without NSX-T.

Please send any feedback you have to pks-feedback@pivotal.io.

Load Balancers in PKS

Page last updated:

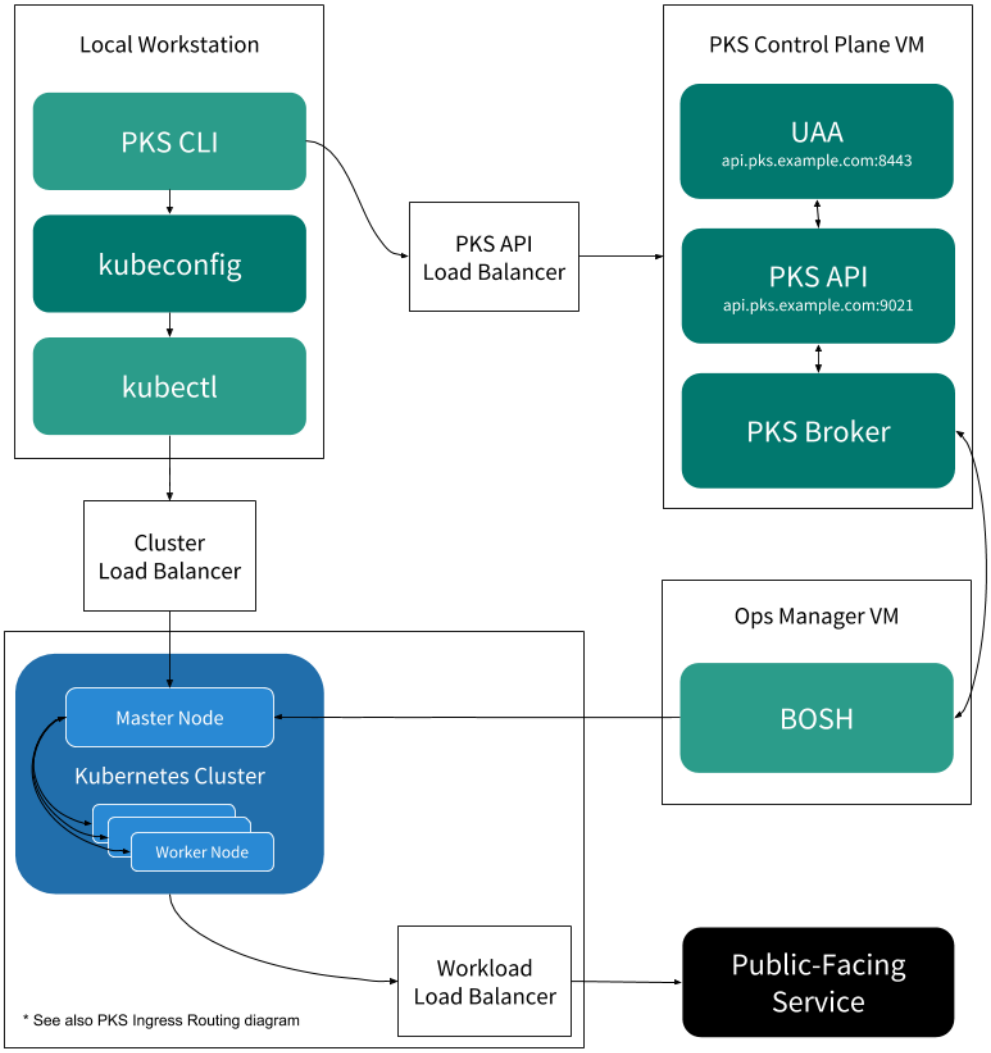
This topic describes the types of load balancers that are used in Pivotal Container Service (PKS) deployments. Load balancers differ by the type of deployment.

Load Balancers in PKS Deployments without NSX-T

For PKS deployments on GCP, AWS, or vSphere without NSX-T, you can configure load balancers for the following:

- **PKS API**: Configuring this load balancer allows you to run PKS Command Line Interface (PKS CLI) commands from your local workstation.
- **Kubernetes Clusters**: Configuring a load balancer for each new cluster allows you to run Kubernetes CLI (kubectl) commands on the cluster.
- **Workloads**: Configuring a load balancer for your application workloads allows external access to the services that run on your cluster.

The following diagram shows where each of the above load balancers can be used within your PKS deployment on GCP, AWS, or on vSphere without NSX-T:



If you use either vSphere without NSX-T or GCP, you are expected to create your own load balancers within your cloud provider console. If your cloud provider does not offer load balancing, you can use any external TCP or HTTPS load balancer of your choice.

About the PKS API Load Balancer

For PKS deployments on GCP, AWS, and on vSphere without NSX-T, the load balancer for the PKS API allows you to access the PKS API from outside the network. For example, configuring a load balancer for the PKS API allows you to run PKS CLI commands from your local workstation.

For information about configuring the PKS API load balancer, see the Configure External Load Balancer section of *Installing PKS* for your IaaS.

About Kubernetes Cluster Load Balancers

For PKS deployments on GCP, AWS, and on vSphere without NSX-T, when you create a cluster, you must configure external access to the cluster by creating an external TCP or HTTPS load balancer. The load balancer allows the Kubernetes CLI to communicate with the cluster.

If you create a cluster in a non-production environment, you can choose not to use a load balancer. To allow kubectl to access the cluster without a load balancer, you can do one of the following:

- Create a DNS entry that points to the cluster's master VM. For example:

```
my-cluster.example.com    A    10.0.0.5
```

- On the workstation where you run kubectl commands, add the master IP address of your cluster and `kubo.internal` to the `/etc/hosts` file. For example:

```
10.0.0.5 kubo.internal
```

For more information about configuring a cluster load balancer, see the following:

- Creating and Configuring a GCP Load Balancer for PKS Clusters
- Creating and Configuring an AWS Load Balancer for PKS Clusters
- Creating and Configuring an Azure Load Balancer for PKS Clusters

About Workload Load Balancers

For PKS deployments on GCP, AWS, and on vSphere without NSX-T, to allow external access to your app, you can either create a load balancer or expose a static port on your workload.

For information about configuring a load balancer for your app workload, see Deploying and Exposing Basic Workloads.

If you use AWS, you must configure routing in the AWS console before you can create a load balancer for your workload. You must create a public subnet in each availability zone (AZ) where you are deploying the workload and tag the public subnet with your cluster's unique identifier.

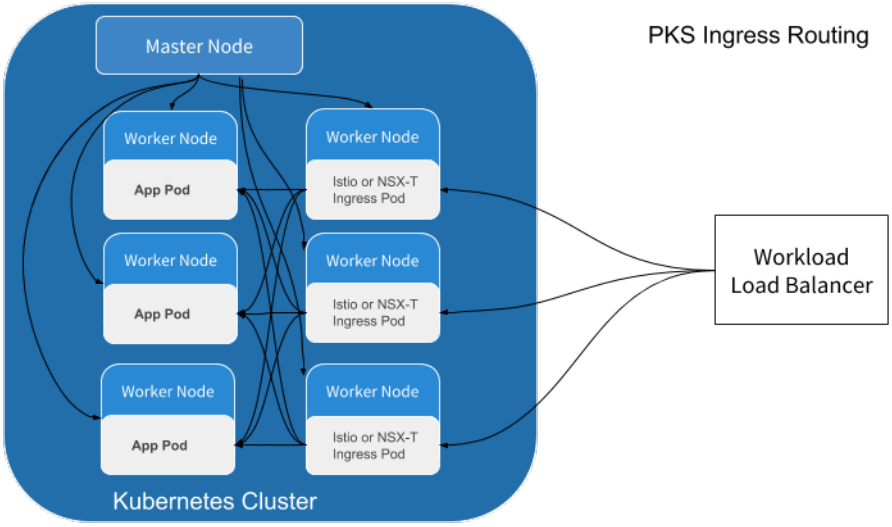
See the AWS Prerequisites section of *Deploying and Exposing Basic Workloads* before you create a workload load balancer.

Deploy Your Workload Load Balancer with an Ingress Controller

A Kubernetes ingress controller sits behind a load balancer, routing HTTP and HTTPS requests from outside the cluster to services within the cluster. Kubernetes ingress resources can be configured to load balance traffic, provide externally reachable URLs to services, and manage other aspects of network traffic.

If you add an ingress controller to your PKS deployment, traffic routing is controlled by the ingress resource rules you define. Pivotal recommends configuring PKS deployments with both a workload load balancer and an ingress controller.

The following diagram shows how the ingress routing can be used within your PKS deployment.



The load balancer on PKS on vSphere with NSX-T is automatically provisioned with Kubernetes ingress resources without the need to deploy and configure an additional ingress controller.

For information about deploying a load balancer configured with ingress routing on GCP, AWS, Azure, and vSphere without NSX-T, see Configuring Ingress Routing. For information about ingress routing on vSphere with NSX-T, see Configuring Ingress Resources and Load Balancer Services.

Load Balancers in PKS Deployments on vSphere with NSX−T

PKS deployments on vSphere with NSX-T do not require a load balancer configured to access the PKS API. They require only a DNAT rule configured so that the PKS API host is accessible. For more information, see Share the PKS Endpoint in *Installing PKS on vSphere with NSX-T Integration*.

NSX-T handles load balancer creation, configuration, and deletion automatically as part of the Kubernetes cluster create, update, and delete process. When a new Kubernetes cluster is created, NSX-T creates and configures a dedicated load balancer tied to it. The load balancer is a shared resource designed to provide efficient traffic distribution to master nodes as well as services deployed on worker nodes. Each application service is mapped to a virtual server instance, carved out from the same load balancer. For more information, see Load Balancing in the NSX-T documentation.

Virtual server instances are created on the load balancer to provide access to the following:

- **Kubernetes API and UI services on a Kubernetes cluster**. This allows requests to be load balanced across multiple master nodes.
- **Ingress controller**. This allows the virtual server instance to dispatch HTTP and HTTPS requests to services associated with Ingress rules.
- `type:loadbalancer` **services**. This allows the server to handle TCP connections or UDP flows toward exposed services.

Load balancers are deployed in high-availability mode so that they are resilient to potential failures and able to recover quickly from critical conditions.

Note: The `NodePort` Service type is not supported for PKS deployments on vSphere with NSX-T. Only `type:LoadBalancer` Services and Services associated with Ingress rules are supported on vSphere with NSX-T.

Resizing Load Balancers

When a new Kubernetes cluster is provisioned using the PKS API, NSX-T creates a dedicated load balancer for that new cluster. By default, the size of the load balancer is set to Small.

With network profiles, you can change the size of the load balancer deployed by NSX-T at the time of cluster creation. For information about network profiles, see Using Network Profiles (NSX-T Only).

For more information about the types of load balancers NSX-T provisions and their capacities, see Scaling Load Balancer Resources in the NSX-T documentation.

Please send any feedback you have to pks-feedback@pivotal.io.

VM Sizing for PKS Clusters

Page last updated:

This topic describes how Pivotal Container Service (PKS) recommends you approach the sizing of VMs for cluster components.

Overview

When you configure plans in the PKS tile, you provide VM sizes for the master and worker node VMs. For more information about configuring plans, see the Plans section of *Installing PKS* for your IaaS:

- vSphere
- vSphere with NSX-T Integration
- Google Cloud Platform (GCP)
- Amazon Web Services (AWS)
- Azure

You select the number of master nodes when you configure the plan.

For worker node VMs, you select the number and size based on the needs of your workload. The sizing of master and worker node VMs is highly dependent on the characteristics of the workload. Adapt the recommendations in this topic based on your own workload requirements.

Master Node VM Size

The master node VM size is linked to the number of worker nodes. The VM sizing shown in the following table is per master node:

Note: If there are multiple master nodes, all master node VMs are the same size. To configure the number of master nodes, see the Plans section of *Installing PKS* for your IaaS.

To customize the size of the Kubernetes master node VM, see Customize Master and Worker Node VM Size and Type.

Number of Workers	CPU	RAM (GB)
1-5	1	3.75
6-10	2	7.5
11-100	4	15
101-250	8	30
251-500	16	60
500+	32	120

Worker Node VM Number and Size

A maximum of 100 pods can run on a single worker node. The actual number of pods that each worker node runs depends on the workload type as well as the CPU and memory requirements of the workload.

To calculate the number and size of worker VMs you require, determine the following for your workload:

- Maximum number of pods you expect to run [`p`]
- Memory requirements per pod [`m`]
- CPU requirements per pod [`c`]

Using the values above, you can calculate the following:

- Minimum number of workers [`W`] = `p / 100`
- Minimum RAM per worker = `m * 100`

- Minimum number of CPUs per worker = `c * 100`

This calculation gives you the minimum number of worker nodes your workload requires. We recommend that you increase this value to account for failures and upgrades.

For example, increase the number of worker nodes by at least one to maintain workload uptime during an upgrade. Additionally, increase the number of worker nodes to fit your own failure tolerance criteria.

The maximum number of worker nodes that you can create for a plan in a PKS-provisioned Kubernetes cluster is set by the **Maximum number of workers on a cluster** field in the **Plans** pane of the PKS tile. To customize the size of the Kubernetes worker node VM, see Customize Master and Worker Node VM Size and Type.

Example Worker Node Requirement Calculation

An example app has the following minimum requirements:

- Number of pods [`p`] = 1000
- RAM per pod [`m`] = 1 GB
- CPU per pod [`c`] = 0.10

To determine how many worker node VMs the app requires, do the following:

1. Calculate the number of workers using `p / 100`:

    ```
    1000/100 = 10 workers
    ```

2. Calculate the minimum RAM per worker using `m * 100`:

    ```
    1 * 100 = 100 GB
    ```

3. Calculate the minimum number of CPUs per worker using `c * 100`:

    ```
    0.10 * 100 = 10 CPUs
    ```

4. For upgrades, increase the number of workers by one:

    ```
    10 workers + 1 worker = 11 workers
    ```

5. For failure tolerance, increase the number of workers by two:

    ```
    11 workers + 2 workers = 13 workers
    ```

In total, this app workload requires 13 workers with 10 CPUs and 100 GB RAM.

Customize Master and Worker Node VM Size and Type

You select the CPU, memory, and disk space for the Kubernetes node VMs from a set list in the PKS tile. Master and worker node VM sizes and types are selected on a per-plan basis. For more information, see the Plans section of the PKS installation topic for your IaaS. For example, Installing PKS on vSphere with NSX-T.

While the list of available node VM types and sizes is extensive, the list may not provide the exact type and size of VM that you want. You can use the Ops Manager API to customize the size and types of the master and worker node VMs. For more information, see How to Create or Remove Custom VM_TYPE Template using the Operations Manager API in the Pivotal Knowledge Base.

Please send any feedback you have to pks-feedback@pivotal.io.

Telemetry

Page last updated:

This topic describes the metrics that the Pivotal Container Service (PKS) tile sends when you enable the VMware Customer Experience Improvement Program (CEIP) or the Pivotal Telemetry Program (Telemetry). You can opt in or opt out of either program in the **Usage Data** pane of the PKS tile.

For more information, see the *Installing PKS* topic for your IaaS:

- vSphere
- vSphere with NSX-T Integration
- Google Cloud Platform (GCP)
- Amazon Web Services (AWS)

Event Envelope Properties

When PKS sends metrics to CEIP or Telemetry, the tile packages the data with the following deployment information:

Property Name	Property Description	Example Data	Added in PKS Version
event	The type of event	create_cluster	v1.1
product_version	PKS tile version	1.2.0-build.40	v1.1
cloud_provider	Cloud provider for the PKS installation	GCP	v1.1
vcenter_id	vCenter ID	00000a11-22bb-3333-4c4c-555566667777	v1.1

Cluster Events

PKS sends metrics for the cluster management events shown in the table below:

Event Name	Event Description	Property Name	Property Description	Added in PKS Version
create_cluster	This event is generated when a user creates a cluster.	**user_id**	A hashed value of the username.	v1.1
		timestamp	The time when the user created the cluster.	v1.1
		plan_name	The name of the PKS plan that was used to create the cluster.	v1.1
		plan_id	The ID of the PKS plan that was used to create the cluster.	v1.1
		cluster_name	The name of the cluster.	v1.1
		cluster_id	The ID of the cluster.	v1.1
		number_of_workers	The number of worker node VMs in the cluster.	v1.1
		number_of_masters	The number of master node VMs in the cluster.	v1.2
resize_cluster	This event is generated when a cluster is resized.	**user_id**	A hashed value of the username.	v1.1
		timestamp	The time when the user created the cluster.	v1.1
		plan_name	The name of the PKS plan that was used to create the cluster.	v1.1
		plan_id	The ID of the PKS plan that was used to create the cluster.	v1.1
		cluster_name	The name of the cluster.	v1.1
		cluster_id	The ID of the cluster.	v1.1
		old_number_of_workers	The number of worker node VMs in the cluster before the resize event.	v1.1
			The number of worker node VMs in the cluster	

		new_number_of_workers	after the resize event.	v1.1
delete_cluster	This event is generated when a user deletes a cluster.	user_id	A hashed value of the username.	v1.1
		timestamp	The time when the user created the cluster.	v1.1
		plan_name	The name of the PKS plan that was used to create the cluster.	v1.1
		plan_id	The ID of the PKS plan that was used to create the cluster.	v1.1
		cluster_name	The name of the cluster.	v1.1
		cluster_id	The ID of the cluster.	v1.1
api_started	This event is generated when the PKS API is started.	authentication_mode	The authentication mode used to access a Kubernetes cluster.	v1.2
		timestamp	The time when the PKS API started.	v1.2

Cluster Metrics

PKS sends both agent metrics and cluster pod metrics for each cluster.

The following table describes cluster agent metrics:

Agent Metric Name	Agent Metric Description	Example	Added in PKS Version
agentid	The unique BOSH-generated deployment name for the cluster.	service-instance_00000a11-22bb-3333-4c4c-555566667777	v1.1
isvrlienabled	If vRealize Log Insight (vRLI) is enabled, this value is true. If vRLI is disabled, this value if false.	true	v1.1
isvropsenabled	If vRealize Operations (vROps) is enabled, this value is true. If vROps is disabled, this value is false.	false	v1.1
iswavefrontenabled	If Wavefront is enabled, this value is true. If Wavefront is disabled, this value is false.	true	v1.1
vcenter_id	This is your vCenter ID.	00000a11-22bb-3333-4c4c-555566667777	v1.1

The following table describes cluster pod metrics:

Cluster Pod Metric Name	Cluster Pod Metric Description	Example	Added in PKS Version
collected_at	This timestamp represents the metric collection time on the agent.	2018-05-31 21:45:27.681 UTC	v1.1
cpu_used	This value represents how much CPU was in use at the time when the event happened.	11412427	v1.1
memory_used	This value represents how much memory was in use at the time when the event happened.	4816896	v1.1
pkst_kubernetesclusterinfo__fk	This value is a foreign key that points to an entry in the *pkst_kubernetesclusterinfo* database.	77777a66-55bb-4444-3c3c-222211110000	v1.1

Please send any feedback you have to pks-feedback@pivotal.io.

PAS and PKS Deployments with Ops Manager

Page last updated:

Ops Manager is a web app that you use to deploy and manage Pivotal Application Service (PAS) and Pivotal Container Service (PKS). This topic explains why Pivotal recommends using separate installations of Ops Manager for PAS and PKS.

For more information about deploying PKS, see Installing PKS.

Security

Ops Manager deploys the PAS and PKS runtime platforms using BOSH. For security reasons, Pivotal does not recommend installing PAS and PKS on the same Ops Manager instance. For even stronger security, Pivotal recommends deploying each Ops Manager instance using a unique cloud provider account.

Tile Configuration and Troubleshooting

Separate installations of Ops Manager allow you to customize and troubleshoot runtime tiles independently. You may choose to configure Ops Manager with different settings for your PAS and PKS deployments.

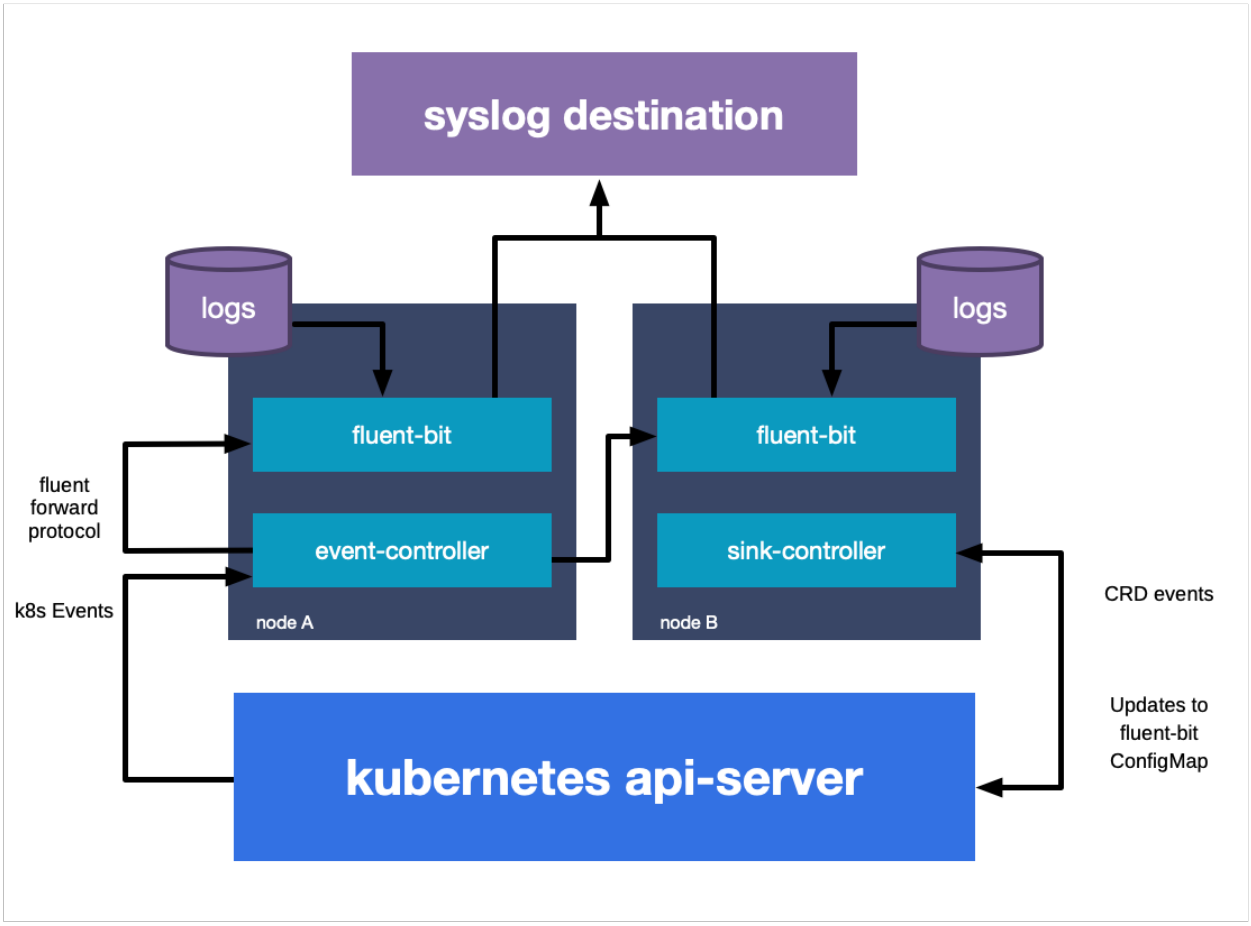
Please send any feedback you have to pks-feedback@pivotal.io.

Sink Architecture in PKS

This topic describes how sinks are implemented in Pivotal Container Service (PKS) deployments.

Sink Architecture Diagram

The following diagram details sink architecture in PKS.



Sink Architecture

Logs are monitored by a set of fluent-bit daemons, which run as a pod on each node.

When sinks are added or removed, all the fluent-bit pods are refreshed with new sink information.

Another pod collects Kubernetes API events and sends them to a fluent-bit pod.

Related Links

For more information on sinks in PKS, see the following topics:

- For information about creating sinks in PKS, see Creating Sink Resources.
- For information about using sinks for monitoring, see Monitoring PKS with Sinks.

Installing PKS

Page last updated:

You can install Pivotal Container Service (PKS) on Amazon Web Services (AWS), Google Cloud Platform (GCP), or vSphere. For installation instructions, see the following:

- vSphere
- vSphere with NSX-T Integration
- GCP
- AWS
- Azure

Please send any feedback you have to pks-feedback@pivotal.io.

vSphere

This topic lists the steps to follow when installing Pivotal Container Service (PKS) on vSphere.

Installing PKS

To install PKS, follow the instructions below:

- Prerequisites and Resource Requirements
- Preparing vSphere Before Deploying PKS
- Deploying Ops Manager on vSphere:
  - Deploying BOSH and Ops Manager v2.3 to vSphere
  - Deploying BOSH and Ops Manager v2.4 to vSphere
- Configuring Ops Manager on vSphere:
  - Configuring BOSH Director v2.3 on vSphere
  - Configuring BOSH Director v2.4 on vSphere
- Installing PKS on vSphere
- (Optional) Integrating VMware Harbor with PKS

Installing the PKS and Kubernetes CLIs

The PKS and Kubernetes CLIs help you interact with your PKS-provisioned Kubernetes clusters and Kubernetes workloads. To install the CLIs, follow the instructions below:

- Installing the PKS CLI
- Installing the Kubernetes CLI

Please send any feedback you have to pks-feedback@pivotal.io.

vSphere Prerequisites and Resource Requirements

Page last updated:

This topic describes the prerequisites and resource requirements for installing Pivotal Container Service (PKS) on vSphere.

For prerequisites and resource requirements for installing PKS on vSphere with NSX-T integration, see vSphere with NSX-T Version Requirements and Hardware Requirements for PKS on vSphere with NSX-T.

PKS supports air-gapped deployments on vSphere with or without NSX-T integration.

You can also configure integration with the Harbor tile, an enterprise-class registry server for container images. For more information, see VMware Harbor Registry in the *Pivotal Partner documentation*.

Prerequisites

Before installing PKS, you must install Ops Manager. You use Ops Manager to install and configure PKS.

To prepare your vSphere environment for installing Ops Manager and PKS, review the sections below and then follow the instructions in Preparing vSphere Before Deploying PKS.

vSphere Version Requirements

PKS on vSphere supports the following vSphere component versions:

Versions	Editions
• VMware vSphere 6.7 U1 EP06 (ESXi670-201901001) – for NSX-T 2.4 • VMware vSphere 6.7 U1 • VMware vSphere 6.7.0 • VMware vSphere 6.5 U2 P03 (ESXi650-201811002) – for NSX-T 2.4 • VMware vSphere 6.5 U2 • VMware vSphere 6.5 U1	• vSphere Enterprise Plus • vSphere with Operations Management Enterprise Plus

Note: VMware vSphere 6.7 is only supported with Ops Manager v2.3.1 or later.

Resource Requirements

Installing Ops Manager and PKS requires the following virtual machines (VMs):

VM	CPU	RAM	Storage
Pivotal Container Service	2	8 GB	16 GB ^*
Pivotal Ops Manager	1	8 GB	160 GB
BOSH Director	2	8 GB	16 GB

Storage Requirements for Large Numbers of Pods

If you expect the cluster workload to run a large number of pods continuously, then increase the size of persistent disk storage allocated to the the Pivotal Container Service VM as follows:

Number of Pods	Storage (Persistent Disk) Requirement ^*
1,000 pods	20 GB
5,000 pods	100 GB

Number of Pods	Storage (Persistent Disk) Requirement ^*
10,000 pods	200 GB
50,000 pods	1,000 GB

Ephemeral VM Resources

Each PKS deployment requires ephemeral VMs during installation and upgrades of PKS. After you deploy PKS, BOSH automatically deletes these VMs. To enable PKS to dynamically create the ephemeral VMs when needed, ensure that the following resources are available in your vSphere infrastructure before deploying PKS:

Ephemeral VM	Number	CPU Cores	RAM	Ephemeral Disk
BOSH Compilation VMs	4	4	4 GB	32 GB

Kubernetes Cluster Resources

Each Kubernetes cluster provisioned through PKS deploys the VMs listed below. If you deploy more than one Kubernetes cluster, you must scale your allocated resources appropriately.

VM	Number	CPU Cores	RAM	Ephemeral Disk	Persistent Disk
master	1 or 3	2	4 GB	8 GB	5 GB
worker	1 or more	2	4 GB	8 GB	50 GB
errand (ephemeral)	1	1	1 GB	8 GB	none

Please send any feedback you have to pks-feedback@pivotal.io.

Firewall Ports and Protocols Requirements for vSphere without NSX-T

Page last updated:

This topic describes the firewall ports and protocols requirements for using Pivotal Container Service (PKS) on vSphere.

Firewalls and security policies are used to filter traffic and limit access in environments with strict inter-network access control policies.

Apps frequently require the ability to pass internal communication between system components on different networks and require one or more conduits through the environment's firewalls. Firewall rules are also required to enable interfacing with external systems such as with enterprise apps or apps and data on the public Internet.

For PKS, Pivotal recommends that you disable security policies that filter traffic between the networks supporting the system. With PKS you should enable access to apps through standard Kubernetes load-balancers and ingress controller types. This enables you to designate specific ports and protocols as a firewall conduit.

For information on ports and protocol requirements for vSphere with NSX-T, see Firewall Ports and Protocols Requirements for vSphere with NSX-T

If you are unable to implement your security policy using the methods described above, refer to the following table, which identifies the flows between system components in a typical PKS deployment.

Note: To control which groups access deploying and scaling your organization's Enterprise PKS-deployed Kubernetes clusters, configure your firewall settings as described on the Operator –> PKS API server lines below.

PKS Ports and Protocols

The following tables list ports and protocols required for network communications between PKS v1.3.6 and later, and vSphere 6.7 and later.

PKS Users Ports and Protocols

The following table lists ports and protocols used for network communication between PKS user interface components.

Source Component	Destination Component	Destination Protocol	Destination Port	Service
Admin/Operator Console	All System Components	TCP	22	ssh
Admin/Operator Console	All System Components	TCP	80	http
Admin/Operator Console	All System Components	TCP	443	https
Admin/Operator Console	Cloud Foundry BOSH Director	TCP	25555	bosh director rest api
Admin/Operator Console	Pivotal Cloud Foundry Operations Manager	TCP	22	ssh
Admin/Operator Console	Pivotal Cloud Foundry Operations Manager	TCP	443	https
Admin/Operator Console	PKS Controller	TCP	9021	pks api server
Admin/Operator Console	vCenter Server	TCP	443	https
Admin/Operator Console	vCenter Server	TCP	5480	vami
Admin/Operator Console	vSphere ESXI Hosts Mgmt. vmknic	TCP	902	ideafarm-door
Admin/Operator and Developer Consoles	Harbor Private Image Registry	TCP	80	http
Admin/Operator and Developer Consoles	Harbor Private Image Registry	TCP	443	https
Admin/Operator and Developer Consoles	Harbor Private Image Registry	TCP	4443	notary
Admin/Operator and Developer Consoles	Kubernetes App Load-Balancer Svc	TCP/UDP	Varies	varies with apps
Admin/Operator and Developer Consoles	Kubernetes Cluster API Server -LB VIP	TCP	8443	httpsca
Admin/Operator and Developer Consoles	Kubernetes Cluster Ingress Controller	TCP	80	http
Admin/Operator and Developer Consoles	Kubernetes Cluster Ingress Controller	TCP	443	https
				kubernetes

Source Component	Destination Component	Destination Protocol	Destination Port	Service
Admin/Operator and Developer Consoles	Kubernetes Cluster Worker Node	TCP/UDP	30000-32767	nodeport
Admin/Operator and Developer Consoles	PKS Controller	TCP	8443	httpsca
All User Consoles (Operator, Developer, Consumer)	Kubernetes App Load-Balancer Svc	TCP/UDP	Varies	varies with apps
All User Consoles (Operator, Developer, Consumer)	Kubernetes Cluster Ingress Controller	TCP	80	http
All User Consoles (Operator, Developer, Consumer)	Kubernetes Cluster Ingress Controller	TCP	443	https
All User Consoles (Operator, Developer, Consumer)	Kubernetes Cluster Worker Node	TCP/UDP	30000-32767	kubernetes nodeport

PKS Core Ports and Protocols

The following table lists ports and protocols used for network communication between core PKS components.

Source Component	Destination Component	Destination Protocol	Destination Port	Service
All System Components	Corporate Domain Name Server	TCP/UDP	53	dns
All System Components	Network Time Server	UDP	123	ntp
All System Components	vRealize LogInsight	TCP/UDP	514/1514	syslog/tls syslog
All System Control Plane Components	AD/LDAP Directory Server	TCP/UDP	389/636	ldap/ldaps
Pivotal Cloud Foundry Operations Manager	Admin/Operator Console	TCP	22	ssh
Pivotal Cloud Foundry Operations Manager	Cloud Foundry BOSH Director	TCP	6868	bosh agent http
Pivotal Cloud Foundry Operations Manager	Cloud Foundry BOSH Director	TCP	8443	httpsca
Pivotal Cloud Foundry Operations Manager	Cloud Foundry BOSH Director	TCP	8844	credhub
Pivotal Cloud Foundry Operations Manager	Cloud Foundry BOSH Director	TCP	25555	bosh director rest api
Pivotal Cloud Foundry Operations Manager	Harbor Private Image Registry	TCP	22	ssh
Pivotal Cloud Foundry Operations Manager	Kubernetes Cluster Master/Etcd Node	TCP	22	ssh
Pivotal Cloud Foundry Operations Manager	Kubernetes Cluster Worker Node	TCP	22	ssh
Pivotal Cloud Foundry Operations Manager	PKS Controller	TCP	22	ssh
Pivotal Cloud Foundry Operations Manager	PKS Controller	TCP	8443	httpsca
Pivotal Cloud Foundry Operations Manager	vCenter Server	TCP	443	https
Pivotal Cloud Foundry Operations Manager	vSphere ESXI Hosts Mgmt. vmknic	TCP	443	https
Cloud Foundry BOSH Director	vCenter Server	TCP	443	https
Cloud Foundry BOSH Director	vSphere ESXI Hosts Mgmt. vmknic	TCP	443	https
BOSH Compilation Job VM	Cloud Foundry BOSH Director	TCP	4222	bosh nats server
BOSH Compilation Job VM	Cloud Foundry BOSH Director	TCP	25250	bosh blobstore
BOSH Compilation Job VM	Cloud Foundry BOSH Director	TCP	25923	health monitor daemon
BOSH Compilation Job VM	Harbor Private Image Registry	TCP	443	https
BOSH Compilation Job VM	Harbor Private Image Registry	TCP	8853	bosh dns health

Source Component	Destination Component	Destination Protocol	Destination Port	Service
PKS Controller	Cloud Foundry BOSH Director	TCP	4222	bosh nats server
PKS Controller	Cloud Foundry BOSH Director	TCP	8443	httpsca
PKS Controller	Cloud Foundry BOSH Director	TCP	25250	bosh blobstore
PKS Controller	Cloud Foundry BOSH Director	TCP	25555	bosh director rest api
PKS Controller	Cloud Foundry BOSH Director	TCP	25923	health monitor daemon
PKS Controller	Kubernetes Cluster Master/Etcd Node	TCP	8443	httpsca
PKS Controller	vCenter Server	TCP	443	https
Harbor Private Image Registry	Cloud Foundry BOSH Director	TCP	4222	bosh nats server
Harbor Private Image Registry	Cloud Foundry BOSH Director	TCP	25250	bosh blobstore
Harbor Private Image Registry	Cloud Foundry BOSH Director	TCP	25923	health monitor daemon
Harbor Private Image Registry	IP NAS Storage Array	TCP	111	nfs rpc portmapper
Harbor Private Image Registry	IP NAS Storage Array	TCP	2049	nfs
Harbor Private Image Registry	Public CVE Source Database	TCP	443	https
kube-system pod/telemetry-agent	PKS Controller	TCP	24224	fluentd out_forward
Kubernetes Cluster Ingress Controller	vCenter Server	TCP	443	https
Kubernetes Cluster Master/Etcd Node	Cloud Foundry BOSH Director	TCP	4222	bosh nats server
Kubernetes Cluster Master/Etcd Node	Cloud Foundry BOSH Director	TCP	25250	bosh blobstore
Kubernetes Cluster Master/Etcd Node	Cloud Foundry BOSH Director	TCP	25923	health monitor daemon
Kubernetes Cluster Master/Etcd Node	Kubernetes Cluster Master/Etcd Node	TCP	2379	etcd clent
Kubernetes Cluster Master/Etcd Node	Kubernetes Cluster Master/Etcd Node	TCP	2380	etcd server
Kubernetes Cluster Master/Etcd Node	Kubernetes Cluster Master/Etcd Node	TCP	8443	httpsca
Kubernetes Cluster Master/Etcd Node	Kubernetes Cluster Master/Etcd Node	TCP	8853	bosh dns health
Kubernetes Cluster Master/Etcd Node	Kubernetes Cluster Worker Node	TCP	4194	cadvisor
Kubernetes Cluster Master/Etcd Node	Kubernetes Cluster Worker Node	TCP	10250	kubelet api
Kubernetes Cluster Master/Etcd Node	Kubernetes Cluster Worker Node	TCP	31194	cadvisor
Kubernetes Cluster Master/Etcd Node	PKS Controller	TCP	8443	httpsca
Kubernetes Cluster Master/Etcd Node	PKS Controller	TCP	8853	bosh dns health
Kubernetes Cluster Master/Etcd Node	vCenter Server	TCP	443	https
Kubernetes Cluster Worker Node	Cloud Foundry BOSH Director	TCP	4222	bosh nats server
Kubernetes Cluster Worker Node	Cloud Foundry BOSH Director	TCP	25250	bosh blobstore
Kubernetes Cluster Worker Node	Cloud Foundry BOSH Director	TCP	25923	health monitor daemon
Kubernetes Cluster Worker Node	Harbor Private Image Registry	TCP	443	https
Kubernetes Cluster Worker Node	Harbor Private Image Registry	TCP	8853	bosh dns health
Kubernetes Cluster Worker Node	IP NAS Storage Array	TCP	111	nfs rpc portmapper
Kubernetes Cluster Worker Node	IP NAS Storage Array	TCP	2049	nfs
Kubernetes Cluster Worker Node	Kubernetes Cluster Master/Etcd Node	TCP	8443	httpsca
Kubernetes Cluster Worker Node	Kubernetes Cluster Master/Etcd Node	TCP	8853	bosh dns health
Kubernetes Cluster Worker Node	Kubernetes Cluster Master/Etcd Node	TCP	10250	kubelet api
pks-system pod/cert-generator	PKS Controller	TCP	24224	fluentd out_forward

Source Component	Destination Component	Destination Protocol	Destination Port	Service
pks-system pod/fluent-bit	PKS Controller	TCP	24224	fluentd out_forward

VMWare Ports and Protocols

The following tables list ports and protocols required for network communication between VMware components.

VMWare Virtual Infrastructure Ports and Protocols

The following table lists ports and protocols used for network communication between VMware virtual infrastructure components.

Source Component	Destination Component	Destination Protocol	Destination Port	Service
vCenter Server	vSphere ESXI Hosts Mgmt. vmknic	TCP	443	https
vCenter Server	vSphere ESXI Hosts Mgmt. vmknic	TCP	8080	http alt
vCenter Server	vSphere ESXI Hosts Mgmt. vmknic	TCP	9080	io filter storage
vSphere ESXI Hosts Mgmt. vmknic	vCenter Server	UDP	902	ideafarm-door
vSphere ESXI Hosts Mgmt. vmknic	vCenter Server	TCP	9084	update manager
vSphere ESXI Hosts Mgmt. vmknic	vSphere ESXI Hosts Mgmt. vmknic	TCP	8182	vsphere ha
vSphere ESXI Hosts Mgmt. vmknic	vSphere ESXI Hosts Mgmt. vmknic	UDP	8182	vsphere ha
vSphere ESXI Hosts vMotion vmknic	vSphere ESXI Hosts vMotion vmknic	TCP	8000	vmotion
vSphere ESXI Hosts IP Storage vmknic	IP NAS Storage Array	TCP	111	nfs rpc portmapper
vSphere ESXI Hosts IP Storage vmknic	IP NAS Storage Array	TCP	2049	nfs
vSphere ESXI Hosts IP Storage vmknic	IP NAS Storage Array	TCP	3260	iscsi
vSphere ESXI Hosts vSAN vmknic	vSphere ESXI Hosts vSAN vmknic	TCP	2233	vsan transport
vSphere ESXI Hosts vSAN vmknic	vSphere ESXI Hosts vSAN vmknic	UDP	12321	unicast agent
vSphere ESXI Hosts vSAN vmknic	vSphere ESXI Hosts vSAN vmknic	UDP	12345	vsan cluster svc
vSphere ESXI Hosts vSAN vmknic	vSphere ESXI Hosts vSAN vmknic	UDP	23451	vsan cluster svc
vSphere ESXI Hosts TEP vmknic	vSphere ESXI Hosts TEP vmknic	UDP	3784	bfd
vSphere ESXI Hosts TEP vmknic	vSphere ESXI Hosts TEP vmknic	UDP	3785	bfd
vSphere ESXI Hosts TEP vmknic	vSphere ESXI Hosts TEP vmknic	UDP	6081	geneve

VMWare Optional Integration Ports and Protocols

The following table lists ports and protocols used for network communication between optional VMware integrations.

Source Component	Destination Component	Destination Protocol	Destination Port	Service
Admin/Operator Console	vRealize Operations Manager	TCP	443	https
vRealize Operations Manager	Kubernetes Cluster API Server -LB VIP	TCP	8443	httpsca
vRealize Operations Manager	PKS Controller	TCP	8443	httpsca
vRealize Operations Manager	Kubernetes Cluster API Server -LB VIP	TCP	8443	httpsca
Admin/Operator Console	vRealize LogInsight	TCP	443	https
Kubernetes Cluster Ingress Controller	vRealize LogInsight	TCP	9000	ingestion api
Kubernetes Cluster Master/Etcd Node	vRealize LogInsight	TCP	9000	ingestion api
Kubernetes Cluster Master/Etcd Node	vRealize LogInsight	TCP	9543	ingestion api -tls
Kubernetes Cluster Worker Node	vRealize LogInsight	TCP	9000	ingestion api
Kubernetes Cluster Worker Node	vRealize LogInsight	TCP	9543	ingestion api -tls
PKS Controller	vRealize LogInsight	TCP	9000	ingestion api
Admin/Operator and Developer Consoles	Wavefront SaaS APM	TCP	443	https
kube-system pod/wavefront-proxy	Wavefront SaaS APM	TCP	443	https

Source Component	Destination Component	Destination Protocol	Destination Port	Service
kube-system pod/wavefront-proxy	Wavefront SaaS API	TCP	8443	httpsca
pks-system pod/wavefront-collector	PKS Controller	TCP	24224	fluentd out_forward
Admin/Operator Console	vRealize Network Insight Platform	TCP	443	https
Admin/Operator Console	vRealize Network Insight Proxy	TCP	22	ssh
vRealize Network Insight Proxy	Kubernetes Cluster API Server -LB VIP	TCP	8443	httpsca
vRealize Network Insight Proxy	PKS Controller	TCP	8443	httpsca
vRealize Network Insight Proxy	PKS Controller	TCP	9021	pks api server

Please send any feedback you have to pks-feedback@pivotal.io.

Preparing vSphere Before Deploying PKS

Page last updated:

This topic describes how to prepare your vSphere environment before deploying Pivotal Container Service (PKS).

Overview

Before you install PKS on vSphere **without** NSX-T integration, you must prepare your vSphere environment by creating the required service accounts and configuring DNS for the PKS API endpoint.

You must create the following service accounts in vSphere:

- **Master Node Service Account** for the Kubernetes master node VMs.
- **BOSH/Ops Manager Service Account** for BOSH Director operations.

WARNING: The PKS Master Node and BOSH/Ops Manager service accounts must be two separate accounts.

After creating the Master Node and BOSH/Ops Manager service accounts you must grant the accounts privileges in vSphere:

- **Master Node Service Account**: Kubernetes master node VMs require storage permissions to create load balancers and attach persistent disks to pods. Creating a custom role for this service account allows vSphere to apply the same privileges to all Kubernetes master node VMs in your PKS installation.
- **BOSH/Ops Manager Service Account**: BOSH Director requires permissions to create VMs. You can apply privileges directly to this service account without creating a role. You can also apply the default VMware Administrator System Role to this service account to achieve the appropriate permission level.

Pivotal recommends configuring each service account with the least permissive privileges and unique credentials.

Note: If your Kubernetes clusters span multiple vCenters, you must set the service account privileges correctly in each vCenter.

To prepare your vSphere environment, do the following:

1. Create the Master Node Service Account
2. Grant Storage Permissions
3. Create the BOSH/Ops Manager Service Account
4. Grant Permissions to the BOSH/Ops Manager Service Account
5. Configure DNS for the PKS API

Prerequisites

Before you prepare your vSphere environment, you must fulfill the prerequisites in vSphere Prerequisites and Resource Requirements.

Create the Master Node Service Account

1. From the vCenter console, create a service account for Kubernetes cluster master VMs.
2. Grant the following **Virtual Machine Object** privileges to the service account:

Privilege (UI)	Privilege (API)
Virtual Machine > Configuration > Advanced	VirtualMachine.Configuration.Advanced
Virtual Machine > Configuration > Settings	VirtualMachine.Configuration.Settings

Grant Storage Permissions

Kubernetes master node VM service accounts require the following:

- Read access to the folder, host, and datacenter of the cluster node VMs
- Permission to create and delete VMs within the resource pool where PKS is deployed

Grant these permissions to the master node service account based on your storage configuration using one of the procedures below:

- Static Only Persistent Volume Provisioning
- Dynamic Persistent Volume Provisioning (with Storage Policy-Based Volume Placement)
- Dynamic Persistent Volume Provisioning (without Storage Policy-Based Volume Placement)

For more information about vSphere storage configurations, see vSphere Storage for Kubernetes in the VMware vSphere documentation.

Static Only Persistent Volume Provisioning

To configure your Kubernetes master node service account using static only Persistent Volume (PV) provisioning, do the following:

1. Create a custom role that allows the service account to manage Kubernetes node VMs. Give this role a name. For example, `manage-k8s-node-vms`. For more information about custom roles in vCenter, see Create a Custom Role in the VMware vSphere documentation.

    a. Grant the following privileges at the **VM Folder** level using either the vCenter UI or API:

Privilege (UI)	Privilege (API)
Virtual Machine > Configuration > Add existing disk	VirtualMachine.Config.AddExistingDisk
Virtual Machine > Configuration > Add new disk	VirtualMachine.Config.AddNewDisk
Virtual Machine > Configuration > Add or remove device	VirtualMachine.Config.AddRemoveDevice
Virtual Machine > Configuration > Remove disk	VirtualMachine.Config.RemoveDisk

    b. Select the **Propagate to Child Objects** checkbox.

2. (Optional) Create a custom role that allows the service account to manage Kubernetes volumes. Give this role a name. For example, `manage-k8s-volumes`.

    Note: This role is required if you create a Persistent Volume Claim (PVC) to bind with a statically provisioned PV, and the reclaim policy is set to delete. When the PVC is deleted, the statically provisioned PV is also deleted.

    a. Grant the following privilege at the **Datastore** level using either the vCenter UI or API:

Privilege (UI)	Privilege (API)
Datastore > Low level file operations	Datastore.FileManagement

    b. Clear the **Propagate to Child Objects** checkbox.

3. Grant the service account the existing **Read-only** role. This role includes the following privileges at the **vCenter, Datacenter, Datastore Cluster,** and **Datastore Storage Folder** levels:

Privilege (UI)	Privilege (API)
Read-only	System.Anonymous
	System.Read
	System.View

4. Continue to Create the BOSH/Ops Manager Service Account.

Dynamic Persistent Volume Provisioning (with Storage Policy-Based Volume Placement)

To configure your Kubernetes master node service account using dynamic PV provisioning **with** storage policy-based placement, do the following:

1. Create a custom role that allows the service account to manage Kubernetes node VMs. Give this role a name. For example, `manage-k8s-node-vms`. For more information about custom roles in vCenter, see Create a Custom Role in the VMware vSphere documentation.

a. Grant the following privileges at the **Cluster, Hosts,** and **VM Folder** levels using either the vCenter UI or API:

Privilege (UI)	Privilege (API)
Virtual Machine > Resource > Assign virtual machine to resource pool	Resource.AssignVMToPool
Virtual Machine > Configuration > Add existing disk	VirtualMachine.Config.AddExistingDisk
Virtual Machine > Configuration > Add new disk	VirtualMachine.Config.AddNewDisk
Virtual Machine > Configuration > Add or remove device	VirtualMachine.Config.AddRemoveDevice
Virtual Machine > Configuration > Remove disk	VirtualMachine.Config.RemoveDisk
Virtual Machine > Inventory > Create new	VirtualMachine.Inventory.Create
Virtual Machine > Inventory > Remove	VirtualMachine.Inventory.Delete

b. Select the **Propagate to Child Objects** checkbox.

2. Create a custom role that allows the service account to manage Kubernetes volumes. Give this role a name. For example, `manage-k8s-volumes`.

a. Grant the following privilege at the **Datastore** level using either the vCenter UI or API:

Privilege (UI)	Privilege (API)
Datastore > Allocate space	Datastore.AllocateSpace
Datastore > Low level file operations	Datastore.FileManagement

b. Clear the **Propagate to Child Objects** checkbox.

3. Create a custom role that allows the service account to read the Kubernetes storage profile. Give this role a name. For example, `k8s-system-read-and-spbm-profile-view`.

a. Grant the following privilege at the **vCenter** level using either the vCenter UI or API:

Privilege (UI)	Privilege (API)
Profile-driven storage view	StorageProfile.View

b. Clear the **Propagate to Child Objects** checkbox.

4. Grant the service account the existing **Read-only** role. This role includes the following privileges at the **vCenter, Datacenter, Datastore Cluster,** and **Datastore Storage Folder** levels:

Privilege (UI)	Privilege (API)
Read-only	System.Anonymous
	System.Read
	System.View

5. Continue to Create the BOSH/Ops Manager Service Account.

Dynamic Volume Provisioning (without Storage Policy-Based Volume Placement)

To configure your Kubernetes master node service account using dynamic PV provisioning **without** storage policy-based placement, do the following:

1. Create a custom role that allows the service account to manage Kubernetes node VMs. Give this role a name. For example, `manage-k8s-node-vms`. For more information about custom roles in vCenter, see Create a Custom Role in the VMware vSphere documentation.

a. Grant the following privileges at the **Cluster, Hosts,** and **VM Folder** levels using either the vCenter UI or API:

Privilege (UI)	Privilege (API)
Virtual Machine > Configuration > Add existing disk	VirtualMachine.Config.AddExistingDisk
Virtual Machine > Configuration > Add new disk	VirtualMachine.Config.AddNewDisk
Virtual Machine > Configuration > Add or remove device	VirtualMachine.Config.AddRemoveDevice
Virtual Machine > Configuration > Remove disk	VirtualMachine.Config.RemoveDisk

b. Select the **Propagate to Child Objects** checkbox.

2. Create a custom role that allows the service account to manage Kubernetes volumes. Give this role a name. For example, `manage-k8s-volumes`.

a. Grant the following privilege at the **Datastore** level using either the vCenter UI or API:

Privilege (UI)	Privilege (API)
Datastore > Allocate space	Datastore.AllocateSpace
Datastore > Low level file operations	Datastore.FileManagement

b. Clear the **Propagate to Child Objects** checkbox.

3. Grant the service account the existing **Read-only** role. This role includes the following privileges at the **vCenter, Datacenter, Datastore Cluster,** and **Datastore Storage Folder** levels:

Privilege (UI)	Privilege (API)
Read-only	System.Anonymous
	System.Read
	System.View

Create the BOSH/Ops Manager Service Account

1. From the vCenter console, create the BOSH/Ops Manager Service Account.
2. If you are deploying both PAS and PKS within the same vSphere environment, create an additional BOSH/Ops Manager Service Account, so that there is one account for PAS and a separate account for PKS.

Grant Permissions to the BOSH/Ops Manager Service Account

There are two options for granting permissions to the BOSH/Ops Manager Service Account(s):

- Grant minimal permissions. Grant each BOSH/Ops Manager Service Account the minimum required permissions as described in vSphere Service Account Requirements.
- Grant Administrator Role permissions. Apply the default VMware Administrator Role to each BOSH/Ops Manager Service Account as described in vCenter Server System Roles.

> **Warning**: Applying the VMware Administrator Role to the BOSH/Ops Manager Service Account grants the account more privileges than are required. For optimal security always use the least privileged account.

Configure DNS for the PKS API

Navigate to your DNS provider and create an entry for a fully qualified domain name (FQDN) within your system domain. For example, `api.pks.example.com`.

When you configure the PKS tile, enter this FQDN in the **PKS API** pane.

After you deploy PKS, you map the IP address of the PKS API to this FQDN. You can then use this FQDN to access the PKS API from your local system.

Next Steps

After you complete the instructions provided in this topic, install one of the following:

- Pivotal Ops Manager v2.3.1 or later
- Pivotal Ops Manager v2.4.x

> **Note:** You use Ops Manager to install and configure PKS. Each version of Ops Manager supports multiple versions of PKS. To confirm that your Ops Manager version supports the version of PKS that you install, see PKS Release Notes.

To install an Ops Manager version that is compatible with the PKS version you intend to use, follow the instructions in the corresponding version of the Ops Manager documentation.

Version	
Ops Manager v2.3	• Deploying BOSH and Ops Manager to vSphere

	• Configuring BOSH Director on vSphere
Ops Manager v2.4	• Deploying BOSH and Ops Manager to vSphere • Configuring BOSH Director on vSphere

Please send any feedback you have to pks-feedback@pivotal.io.

Installing PKS on vSphere

Page last updated:

This topic describes how to install and configure Pivotal Container Service (PKS) on vSphere.

Prerequisites

Before performing the procedures in this topic, you must have deployed and configured Ops Manager. For more information, see vSphere Prerequisites and Resource Requirements.

If you use an instance of Ops Manager that you configured previously to install other runtimes, perform the following steps before you install PKS:

1. Navigate to Ops Manager.
2. Open the **Director Config** pane.
3. Select the **Enable Post Deploy Scripts** checkbox.
4. Click the **Installation Dashboard** link to return to the Installation Dashboard.
5. Click **Review Pending Changes**. Select all products you intend to deploy and review the changes. For more information, see Reviewing Pending Product Changes.
6. Click **Apply Changes**.

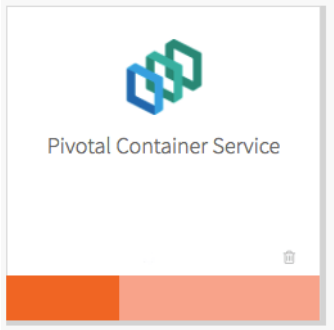
Step 1: Install PKS

To install PKS, do the following:

1. Download the product file from Pivotal Network.
2. Navigate to `https://YOUR-OPS-MANAGER-FQDN/` in a browser to log in to the Ops Manager Installation Dashboard.
3. Click **Import a Product** to upload the product file.
4. Under **Pivotal Container Service** in the left column, click the plus sign to add this product to your staging area.

Step 2: Configure PKS

Click the orange **Pivotal Container Service** tile to start the configuration process.



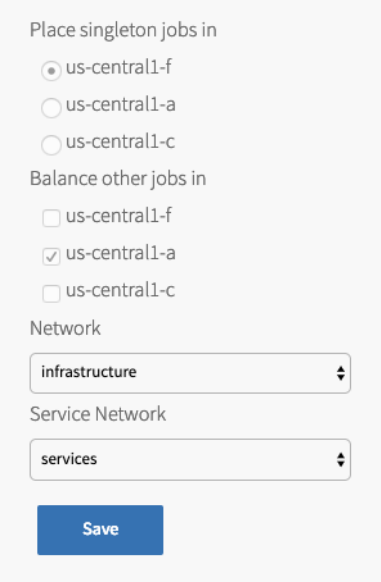
WARNING: When you configure the PKS tile, do not use spaces in any field entries. This includes spaces between characters as well as leading and trailing spaces. If you use a space in any field entry, the deployment of PKS fails.

Assign AZs and Networks

Perform the following steps:

1. Click **Assign AZs and Networks**.
2. Select the availability zone (AZ) where you want to deploy the PKS API VM as a singleton job.

> **Note**: You must select an additional AZ for balancing other jobs before clicking **Save**, but this selection has no effect in the current version of PKS.

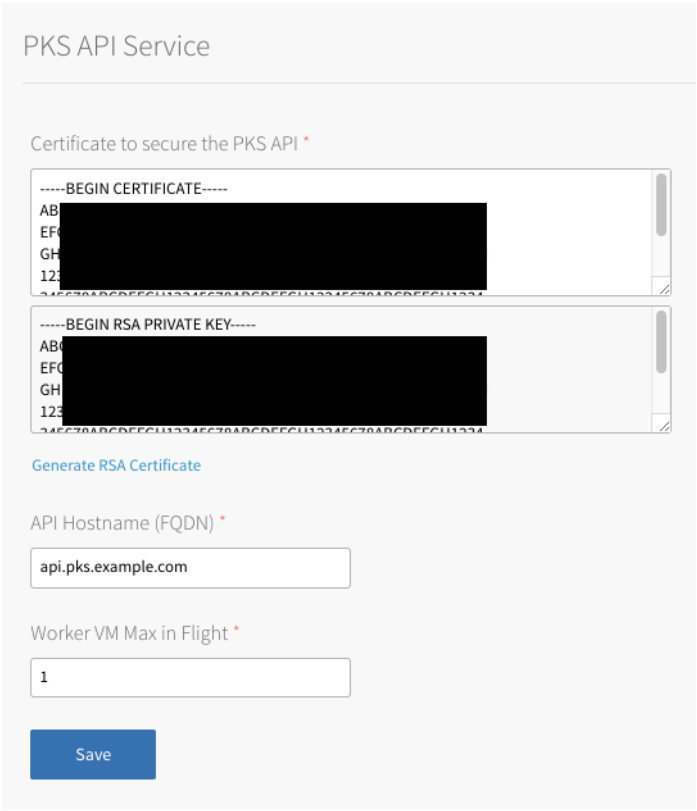


3. Under **Network**, select the infrastructure subnet that you created for the PKS API VM.
4. Under **Service Network**, select the services subnet that you created for Kubernetes cluster VMs.
5. Click **Save**.

PKS API

Perform the following steps:

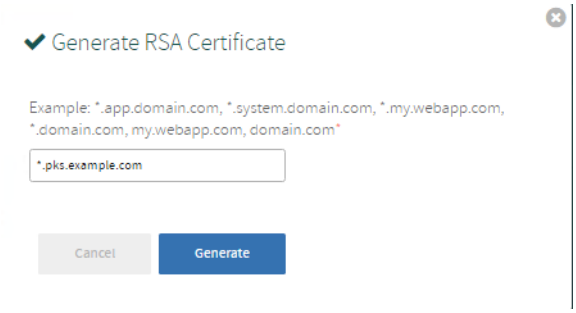
1. Click **PKS API**.
2. Under **Certificate to secure the PKS API**, provide your own certificate and private key pair.



The certificate that you supply should cover the domain that routes to the PKS API VM with TLS termination on the ingress.

If you do not have a certificate and private key pair, PKS can generate one for you. To generate a certificate, do the following:

a. Select the **Generate RSA Certificate** link.
b. Enter the domain for your API hostname. This can be a standard FQDN or a wildcard domain.
c. Click **Generate**.



3. Under **API Hostname (FQDN)**, enter the FQDN that you registered to point to the PKS API load balancer, such as `api.pks.example.com`. To retrieve the public IP address or FQDN of the PKS API load balancer, log in to your IaaS console.

4. Under **Worker VM Max in Flight**, enter the maximum number of non-canary worker instances to create or resize in parallel within an availability zone.

   This field sets the `max_in_flight` variable, which limits how many instances of a component can start simultaneously when a cluster is created or resized. The variable defaults to `1`, which means that only one component starts at a time.

5. Click **Save**.

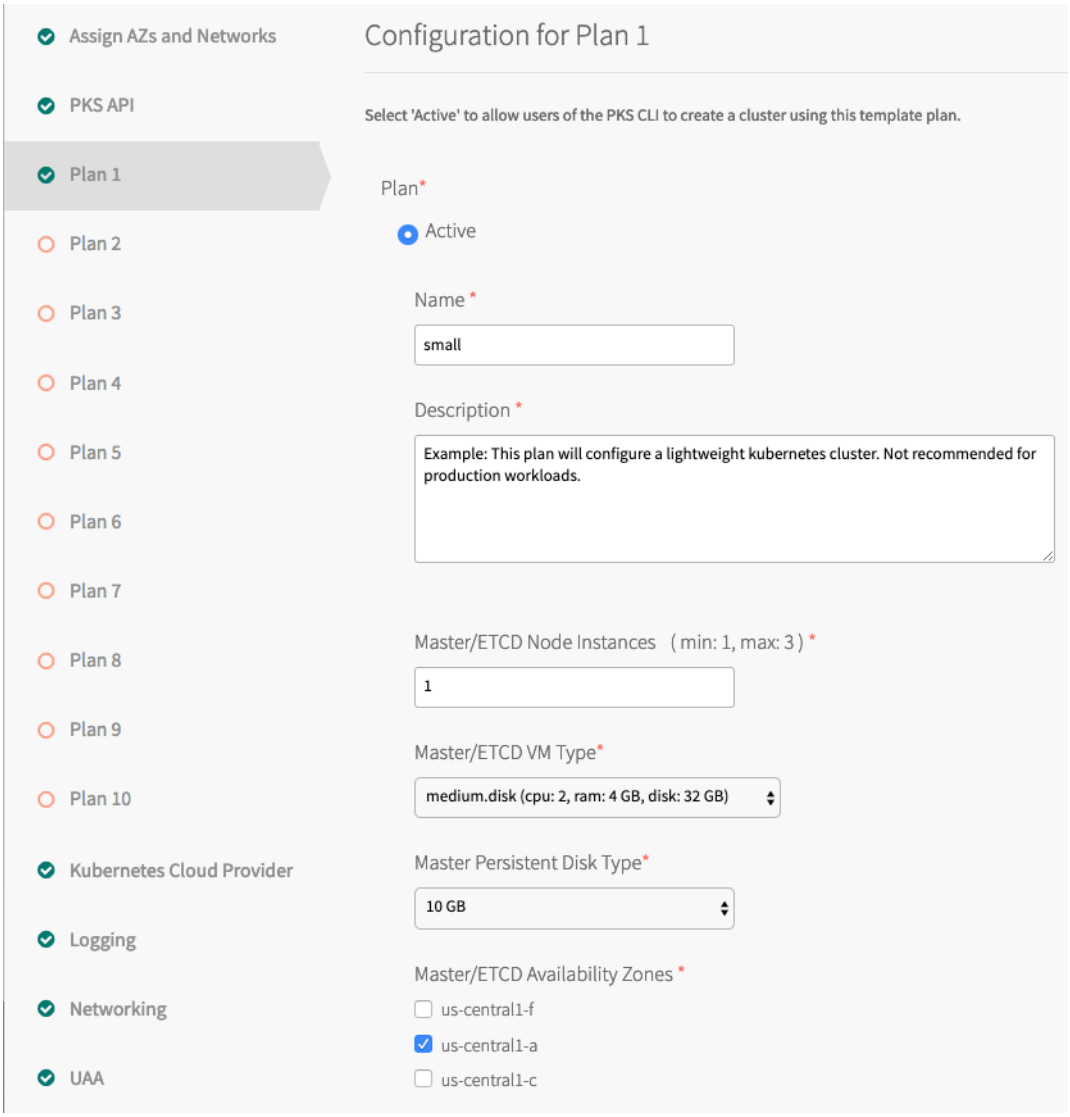
Plans

To activate a plan, perform the following steps:

1. Click the plan that you want to activate.

> **Note**: A plan defines a set of resource types used for deploying clusters. You can configure up to ten plans. You must configure **Plan 1**.

2. Select **Active** to activate the plan and make it available to developers deploying clusters.



3. Under **Name**, provide a unique name for the plan.

4. Under **Description**, edit the description as needed. The plan description appears in the Services Marketplace, which developers can access by using PKS CLI.

5. Under **Master/ETCD Node Instances**, select the default number of Kubernetes master/etcd nodes to provision for each cluster. You can enter either `1` or `3`.

> **Note**: If you deploy a cluster with multiple master/etcd node VMs, confirm that you have sufficient hardware to handle the increased load on disk write and network traffic. For more information, see Hardware recommendations in the etcd documentation.
>
> In addition to meeting the hardware requirements for a multi-master cluster, we recommend configuring monitoring for etcd to monitor disk latency, network latency, and other indicators for the health of the cluster. For more information, see Monitoring Master/etcd Node VMs.

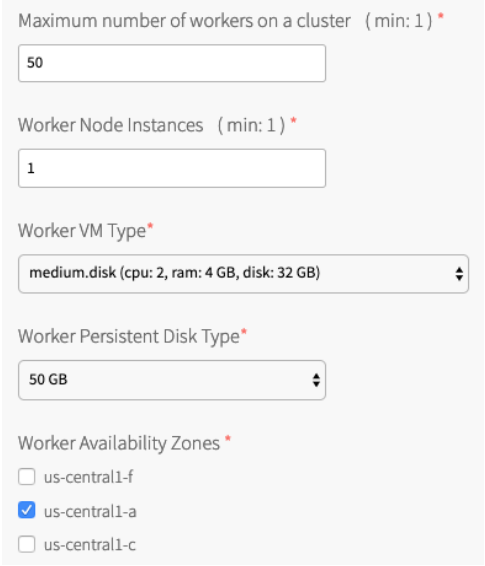
> **WARNING**: To change the number of master/etcd nodes for a plan, you must ensure that no existing clusters use the plan. PKS does not support changing the number of master/etcd nodes for plans with existing clusters.

6. Under **Master/ETCD VM Type**, select the type of VM to use for Kubernetes master/etcd nodes. For more information, including master node VM customization options, see the Master Node VM Size section of *VM Sizing for PKS Clusters*.

7. Under **Master Persistent Disk Type**, select the size of the persistent disk for the Kubernetes master node VM.

8. Under **Master/ETCD Availability Zones**, select one or more AZs for the Kubernetes clusters deployed by PKS. If you select more than one AZ, PKS deploys the master VM in the first AZ and the worker VMs across the remaining AZs.

9. Under **Maximum number of workers on a cluster**, set the maximum number of Kubernetes worker node VMs that PKS can deploy for each cluster. Enter any whole number in this field.



10. Under **Worker Node Instances**, select the default number of Kubernetes worker nodes to provision for each cluster.

    If the user creating a cluster with the PKS CLI does not specify a number of worker nodes, the cluster is deployed with the default number set in this field. This value cannot be greater than the maximum worker node value you set in the previous field. For more information about creating clusters, see Creating Clusters.

    For high availability, create clusters with a minimum of three worker nodes, or two per AZ if you intend to use PersistentVolumes (PVs). For example, if you deploy across three AZs, you should have six worker nodes. For more information about PVs, see PersistentVolumes in *Maintaining Workload Uptime*. Provisioning a minimum of three worker nodes, or two nodes per AZ is also recommended for stateless workloads.

    If you later reconfigure the plan to adjust the default number of worker nodes, the existing clusters that have been created from that plan are not automatically upgraded with the new default number of worker nodes.

11. Under **Worker VM Type**, select the type of VM to use for Kubernetes worker node VMs. For more information, including worker node VM customization options, see the Worker Node VM Number and Size section of *VM Sizing for PKS Clusters*.

    Note: If you install PKS in an NSX-T environment, we recommend that you select a **Worker VM Type** with a minimum disk size of 16 GB. The disk space provided by the default `medium` Worker VM Type is insufficient for PKS with NSX-T.

12. Under **Worker Persistent Disk Type**, select the size of the persistent disk for the Kubernetes worker node VMs.

13. Under **Worker Availability Zones**, select one or more AZs for the Kubernetes worker nodes. PKS deploys worker nodes equally across the AZs you select.

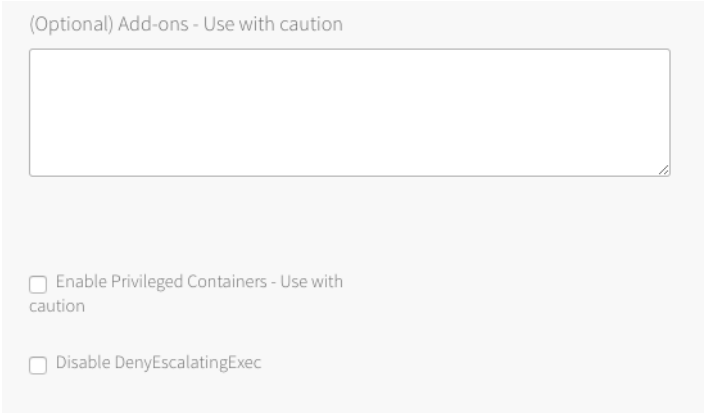
14. Under **Kubelet customization - system-reserved**, enter resource values that Kubelet can use to reserve resources for system daemons. For example, `memory=250Mi, cpu=150m`. For more information about system-reserved values, see the Kubernetes documentation.

15. Under **Kubelet customization - eviction-hard**, enter threshold limits that Kubelet can use to evict pods when they exceed the limit. Enter limits in the format `EVICTION-SIGNAL=QUANTITY`. For example, `memory.available=100Mi, nodefs.available=10%, nodefs.inodesFree=5%`. For more information about eviction thresholds, see the Kubernetes documentation.

    WARNING: Use the Kubelet customization fields with caution. If you enter values that are invalid or that exceed the limits the system supports, Kubelet might fail to start. If Kubelet fails to start, you cannot create clusters.

16. Under **Errand VM Type**, select the size of the VM that contains the errand. The smallest instance possible is sufficient, as the only errand running on this VM is the one that applies the **Default Cluster App** YAML configuration.

17. (Optional) Under **(Optional) Add-ons - Use with caution**, enter additional YAML configuration to add custom workloads to each cluster in this plan. You can specify multiple files using `---` as a separator. For more information, see Adding Custom Workloads.



18. (Optional) To allow users to create pods with privileged containers, select the **Enable Privileged Containers - Use with caution** option. For more information, see Pods in the Kubernetes documentation.
19. (Optional) To disable the admission controller, select the **Disable DenyEscalatingExec** checkbox. If you select this option, clusters in this plan can create security vulnerabilities that may impact other tiles. Use this feature with caution.
20. Click **Save**.

To deactivate a plan, perform the following steps:

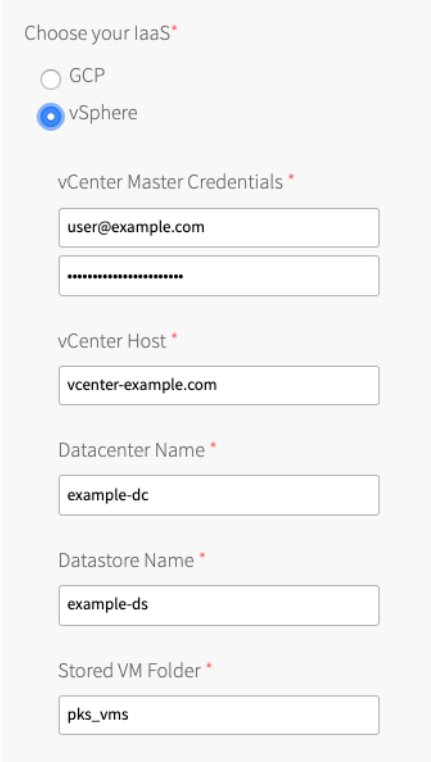
1. Click the plan that you want to deactivate.
2. Select **Inactive**.
3. Click **Save**.

Kubernetes Cloud Provider

In the procedure below, you use credentials for vCenter master VMs. You must have provisioned the service account with the correct permissions. For more information, see Create the Master Node Service Account in *Preparing vSphere Before Deploying PKS*.

To configure your Kubernetes cloud provider settings, follow the procedure below:

1. Click **Kubernetes Cloud Provider**.
2. Under **Choose your IaaS**, select **vSphere**.
3. Ensure the values in the following procedure match those in the **vCenter Config** section of the Ops Manager tile.



a. Enter your **vCenter Master Credentials**. Enter the username using the format `user@example.com`. For more information about the master node service account, see Preparing to Deploy PKS on vSphere.
b. Enter your **vCenter Host**. For example, `vcenter-example.com`.
c. Enter your **Datacenter Name**. For example, `example-dc`.
d. Enter your **Datastore Name**. For example, `example-ds`.
e. Enter the **Stored VM Folder** so that the persistent stores know where to find the VMs. To retrieve the name of the folder, navigate to your BOSH Director tile, click **vCenter Config**, and locate the value for **VM Folder**. The default folder name is `pcf_vms`.
f. Click **Save**.

Note: The value for the **Datastore Name** field is intended to be a single datastore that is the default target. This field should not include a list of BOSH Job/VMDK datastores. The default datastore is used if the Kubernetes cluster `StorageClass` does not define a `StoragePolicy`. For more information, see PersistentVolume Storage Options on vSphere.

Note: For multi-AZ and multi-cluster environments, we recommend using a shared datastore that is available to each vSphere cluster, as opposed to a datastore that is local to a single cluster. For more information, see PersistentVolume Storage Options on vSphere.

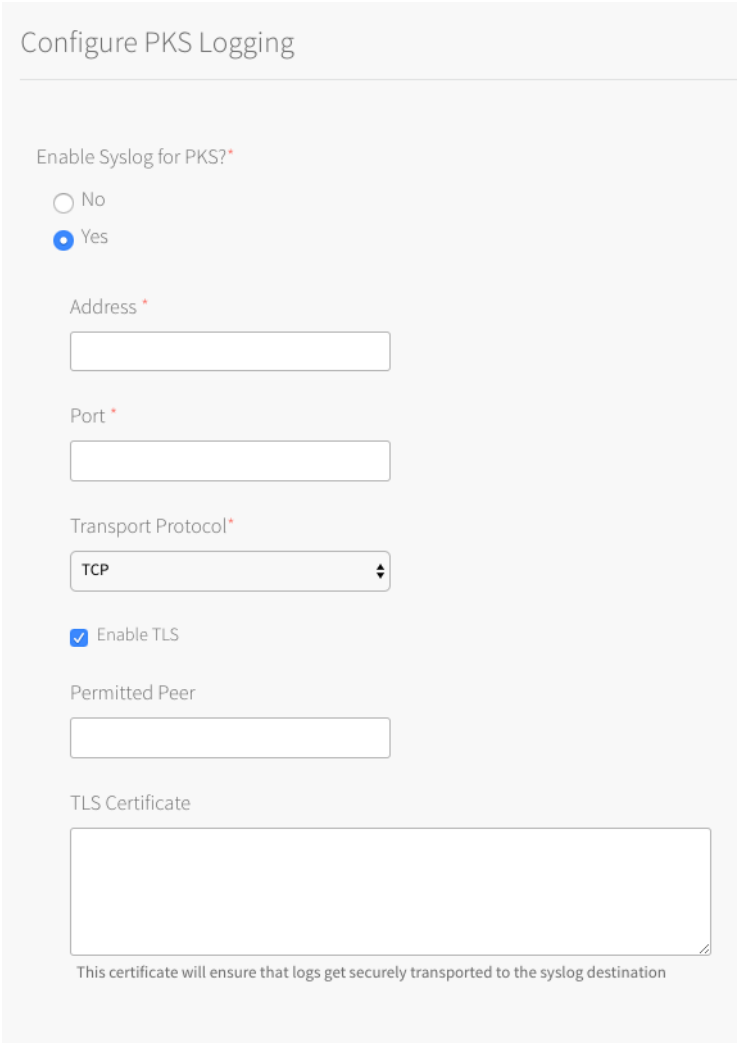
(Optional) Logging

You can designate an external syslog endpoint for forwarded BOSH-deployed VM logs.

In addition, you can enable sink resources to collect PKS cluster and namespace log messages.

To configure logging in PKS, do the following:

1. Click **Logging**.
2. To enable syslog forwarding for BOSH-deployed VM logs, select **Yes**.



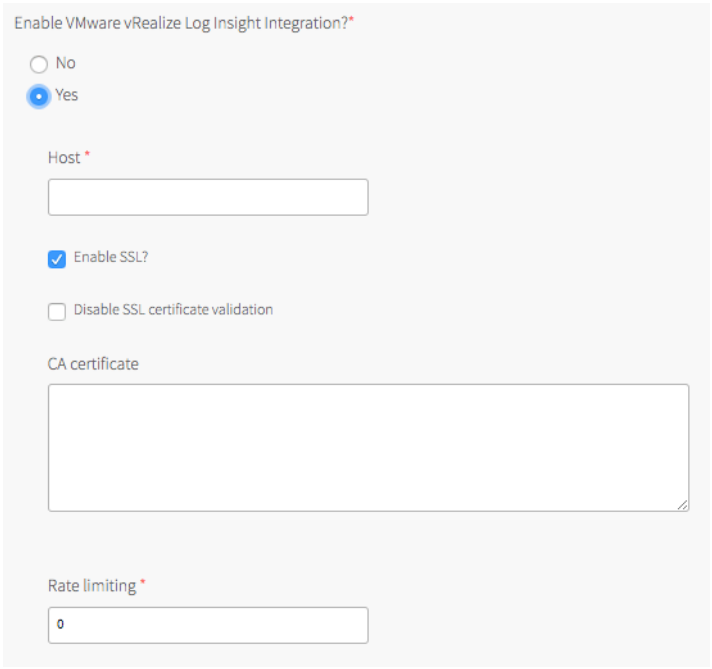
3. Under **Address**, enter the destination syslog endpoint.
4. Under **Port**, enter the destination syslog port.
5. Select a transport protocol for log forwarding.
6. (Optional) Pivotal strongly recommends that you enable TLS encryption when forwarding logs as they may contain sensitive information. For example, these logs may contain cloud provider credentials. To enable TLS, perform the following steps:
    a. Under **Permitter Peer**, provide the accepted fingerprint (SHA1) or name of remote peer. For example, `*.YOUR-LOGGING-SYSTEM.com`.
    b. Under **TLS Certificate**, provide a TLS certificate for the destination syslog endpoint.

    > **Note**: You do not need to provide a new certificate if the TLS certificate for the destination syslog endpoint is signed by a Certificate Authority (CA) in your BOSH certificate store.

7. You can manage logs using VMware vRealize Log Insight (vRLI). The integration pulls logs from all BOSH jobs and containers running in the cluster, including node logs from core Kubernetes and BOSH processes, Kubernetes event logs, and POD stdout and stderr.

> **Note**: Before you configure the vRLI integration, you must have a vRLI license and vRLI must be installed, running, and available in your environment. You need to provide the live instance address during configuration. For instructions and additional information, see the vRealize Log Insight documentation.

By default, vRLI logging is disabled. To enable and configure vRLI logging, under **Enable VMware vRealize Log Insight Integration?**, select **Yes** and



then perform the following steps:

a. Under **Host**, enter the IP address or FQDN of the vRLI host.
b. (Optional) Select the **Enable SSL?** checkbox to encrypt the logs being sent to vRLI using SSL.
c. Choose one of the following SSL certificate validation options:

- To skip certificate validation for the vRLI host, select the **Disable SSL certificate validation** checkbox. Select this option if you are using a self-signed certificate in order to simplify setup for a development or test environment.

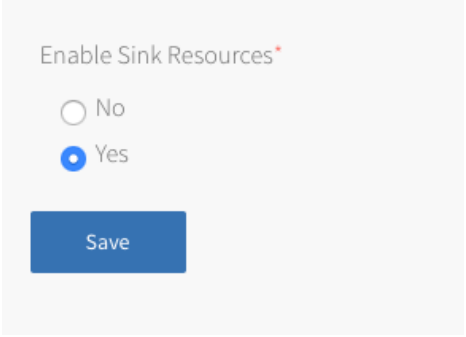
  Note: Disabling certificate validation is not recommended for production environments.

- To enable certificate validation for the vRLI host, clear the **Disable SSL certificate validation** checkbox.

d. (Optional) If your vRLI certificate is not signed by a trusted CA root or other well known certificate, enter the certificate in the **CA certificate** field. Locate the PEM of the CA used to sign the vRLI certificate, copy the contents of the certificate file, and paste them into the field. Certificates must be in PEM-encoded format.
e. Under **Rate limiting**, enter a time in milliseconds to change the rate at which logs are sent to the vRLI host. The rate limit specifies the minimum time between messages before the fluentd agent begins to drop messages. The default value (0) means the rate is not limited, which suffices for many deployments.

Note: If your deployment is generating a high volume of logs, you can increase this value to limit network traffic. Consider starting with a lower number, such as 10, and tuning to optimize for your deployment. A large number might result in dropping too many log entries.

8. To enable clusters to drain app logs to sinks using `syslog://`, select the **Enable Sink Resources** checkbox. For more information about using sink resources, see Creating Sink Resources.



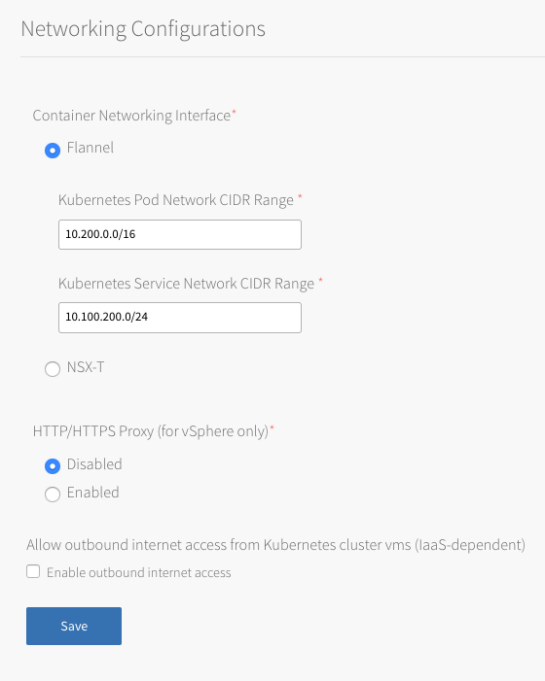
9. Click **Save**. These settings apply to any clusters created after you have saved these configuration settings and clicked **Apply Changes**. If the **Upgrade all clusters errand** has been enabled, these settings are also applied to existing clusters.

Note: The PKS tile does not validate your vRLI configuration settings. To verify your setup, look for log entries in vRLI.

Networking

To configure networking, do the following:

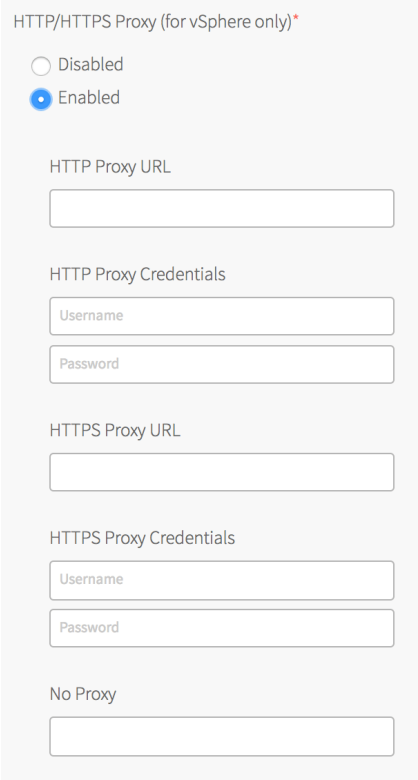
1. Click **Networking**.



2. Under **Container Networking Interface**, select **Flannel**.
3. (Optional) Enter values for **Kubernetes Pod Network CIDR Range** and **Kubernetes Service Network CIDR Range**.
    - Ensure that the CIDR ranges do not overlap and have sufficient space for your deployed services.
    - Ensure that the CIDR range for the **Kubernetes Pod Network CIDR Range** is large enough to accommodate the expected maximum number of pods.
4. (Optional) Configure a global proxy for all outgoing HTTP/HTTPS traffic from your Kubernetes clusters. This setting will not set the proxy for running Kubernetes workloads or pods.

    Production environments can deny direct access to public Internet services and between internal services by placing an HTTP/HTTPS proxy in the network path between Kubernetes nodes and those services.

    If your environment includes HTTP/HTTPS proxies, configuring PKS to use these proxies allows PKS-deployed Kubernetes nodes to access public Internet services and other internal services. Follow the steps below to configure a global proxy for all outgoing HTTP/HTTPS traffic from your Kubernetes clusters:



a. Under **HTTP/HTTPS proxy**, select **Enabled**.
b. Under **HTTP Proxy URL**, enter the URL of your HTTP/HTTPS proxy endpoint. For example, `http://myproxy.com:1234`.
c. (Optional) If your proxy uses basic authentication, enter the username and password under **HTTP Proxy Credentials**.
d. Under **No Proxy**, enter the service network CIDR where your PKS cluster is deployed. List any additional IP addresses or domain names that should bypass the proxy. The **No Proxy** property for vSphere accepts wildcard domains denoted by a prefixed `*.` or `.`, for example `*.example.com` and `.example.com`.

> **Note**: By default, the `.internal`, `10.100.0.0/8`, and `10.200.0.0/8` IP address ranges are not proxied. This allows internal PKS communication.
>
> Do not use the `_` character in the **No Proxy** field. Entering an underscore character in this field can cause upgrades to fail.
>
> Because some jobs in the VMs accept `*.` as a wildcard, while others only accept `.`, we recommend that you define a wildcard domain using both of them. For example, to denote `example.com` as a wildcard domain, add both `*.example.com` and `example.com` to the **No Proxy** property.

5. Under **Allow outbound internet access from Kubernetes cluster vms (IaaS-dependent)**, ignore the **Enable outbound internet access** checkbox.

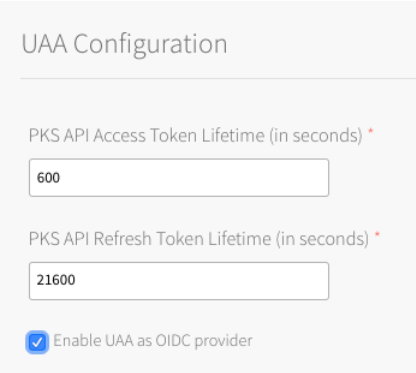
6. Click **Save**.

UAA

To configure the UAA server, do the following:

1. Click **UAA**.

2. Under **PKS API Access Token Lifetime**, enter a time in seconds for the PKS API access token lifetime.



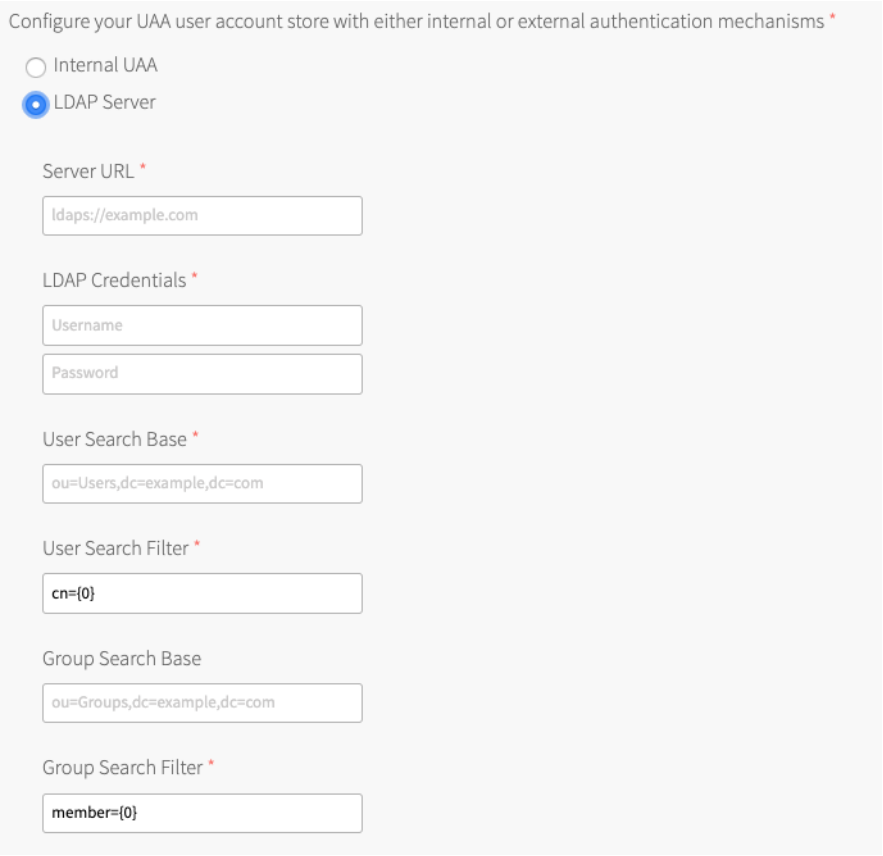
3. Under **PKS API Refresh Token Lifetime**, enter a time in seconds for the PKS API refresh token lifetime.
4. Select one of the following options:
    - To use an internal user account store for UAA, select **Internal UAA**. Click **Save** and continue to (Optional) Monitoring.
    - To use an external user account store for UAA, select **LDAP Server** and continue to Configure LDAP as an Identity Provider.

> **Note**: Selecting **LDAP Server** allows admin users to give cluster access to groups of users. For more information about performing this procedure, see Grant Cluster Access to a Group in *Managing Users in PKS with UAA*.

Configure LDAP as an Identity Provider

To integrate UAA with one or more LDAP servers, configure PKS with your LDAP endpoint information as follows:

1. Under **UAA**, select **LDAP Server**.



2. For **Server URL**, enter the URLs that point to your LDAP server. If you have multiple LDAP servers, separate their URLs with spaces. Each URL must include one of the following protocols:
    - `ldap://` : Use this protocol if your LDAP server uses an unencrypted connection.

- `ldaps://`: Use this protocol if your LDAP server uses SSL for an encrypted connection. To support an encrypted connection, the LDAP server must hold a trusted certificate or you must import a trusted certificate to the JVM truststore.

3. For **LDAP Credentials**, enter the LDAP Distinguished Name (DN) and password for binding to the LDAP server. For example, `cn=administrator,ou=Users,dc=example,dc=com`. If the bind user belongs to a different search base, you must use the full DN.

   Note: We recommend that you provide LDAP credentials that grant read-only permissions on the LDAP search base and the LDAP group search base.

4. For **User Search Base**, enter the location in the LDAP directory tree where LDAP user search begins. The LDAP search base typically matches your domain name.

   For example, a domain named `cloud.example.com` may use `ou=Users,dc=example,dc=com` as its LDAP user search base.

5. For **User Search Filter**, enter a string to use for LDAP user search criteria. The search criteria allows LDAP to perform more effective and efficient searches. For example, the standard LDAP search filter `cn=Smith` returns all objects with a common name equal to `Smith`.

   In the LDAP search filter string that you use to configure PKS, use `{0}` instead of the username. For example, use `cn={0}` to return all LDAP objects with the same common name as the username.

   In addition to `cn`, other common attributes are `mail`, `uid` and, in the case of Active Directory, `sAMAccountName`.

   Note: For information about testing and troubleshooting your LDAP search filters, see Configuring LDAP Integration with Pivotal Cloud Foundry.

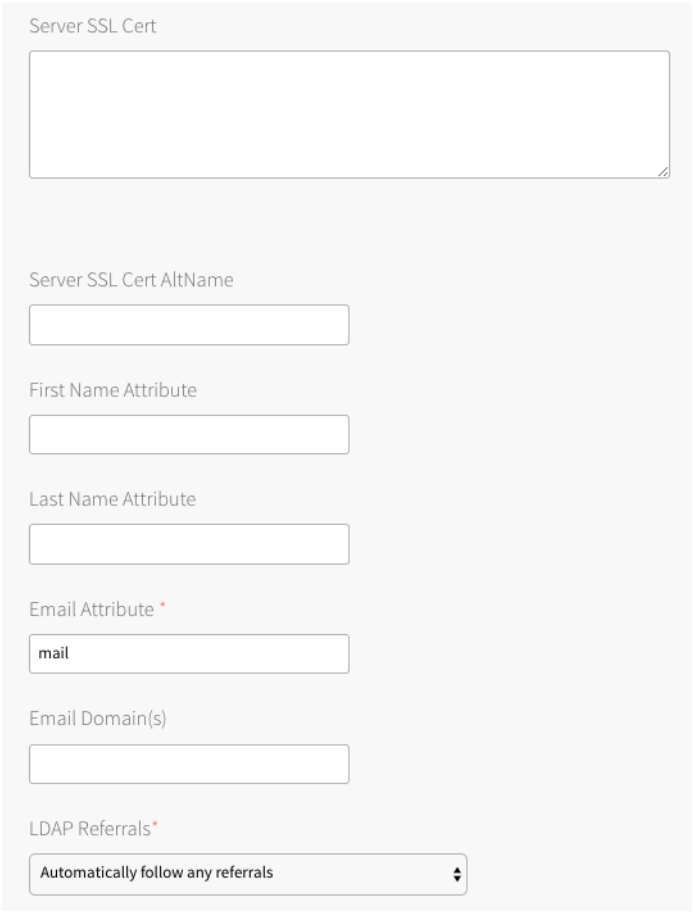
6. For **Group Search Base**, enter the location in the LDAP directory tree where the LDAP group search begins.

   For example, a domain named `cloud.example.com` may use `ou=Groups,dc=example,dc=com` as its LDAP group search base.

   Follow the instructions in the Grant PKS Access to an External LDAP Group section of *Managing Users in PKS with UAA* to map the groups under this search base to roles in PKS.

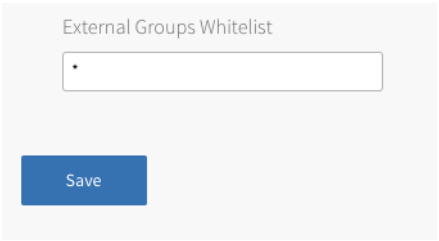
7. For **Group Search Filter**, enter a string that defines LDAP group search criteria. The standard value is `member={0}`.

8. For **Server SSL Cert**, paste in the root certificate from your CA certificate or your self-signed certificate.



9. For **Server SSL Cert AltName**, do one of the following:
    - If you are using `ldaps://` with a self-signed certificate, enter a Subject Alternative Name (SAN) for your certificate.
    - If you are not using `ldaps://` with a self-signed certificate, leave this field blank.
10. For **First Name Attribute**, enter the attribute name in your LDAP directory that contains user first names. For example, `cn`.
11. For **Last Name Attribute**, enter the attribute name in your LDAP directory that contains user last names. For example, `sn`.
12. For **Email Attribute**, enter the attribute name in your LDAP directory that contains user email addresses. For example, `mail`.
13. For **Email Domain(s)**, enter a comma-separated list of the email domains for external users who can receive invitations to Apps Manager.
14. For **LDAP Referrals**, choose how UAA handles LDAP server referrals to other user stores. UAA can follow the external referrals, ignore them without returning errors, or generate an error for each external referral and abort the authentication.
15. For **External Groups Whitelist**, enter a comma-separated list of group patterns which need to be populated in the user's `id_token`. For further information on accepted patterns see the description of the `config.externalGroupsWhitelist` in the OAuth/OIDC Identity Provider Documentation.

    Note: When sent as a Bearer token in the Authentication header, wide pattern queries for users who are members of multiple groups, can cause the size of the `id_token` to extend beyond what is supported by web servers.

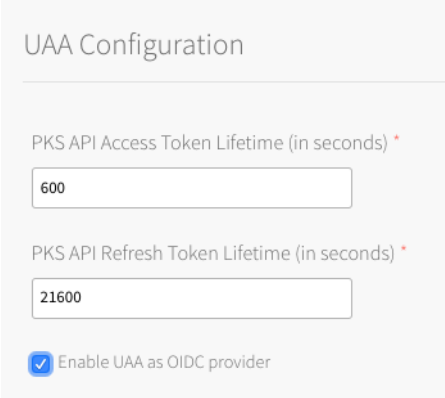


16. Click **Save**.

(Optional) Configure OpenID Connect

You can use OpenID Connect (OIDC) to instruct Kubernetes to verify end-user identities based on authentication performed by an authorization server, such as UAA.

To configure PKS to use OIDC, select **Enable UAA as OIDC provider**. With OIDC enabled, Admin Users can grant cluster-wide access to Kubernetes end users.



For more information about configuring OIDC, see the table below:

Option	Description
OIDC disabled	If you do not enable OIDC, Kubernetes authenticates users against its internal user management system.
OIDC enabled	If you enable OIDC, Kubernetes uses the authentication mechanism that you selected in UAA as follows: • If you selected **Internal UAA**, Kubernetes authenticates users against the internal UAA authentication mechanism. • If you selected **LDAP Server**, Kubernetes authenticates users against the LDAP server.

For additional information about getting credentials with OIDC configured, see Retrieve Cluster Credentials in *Retrieving Cluster Credentials and Configuration*.

Note: When you enable OIDC, existing PKS-provisioned Kubernetes clusters are upgraded to use OIDC. This invalidates your kubeconfig files. You must regenerate the files for all clusters.

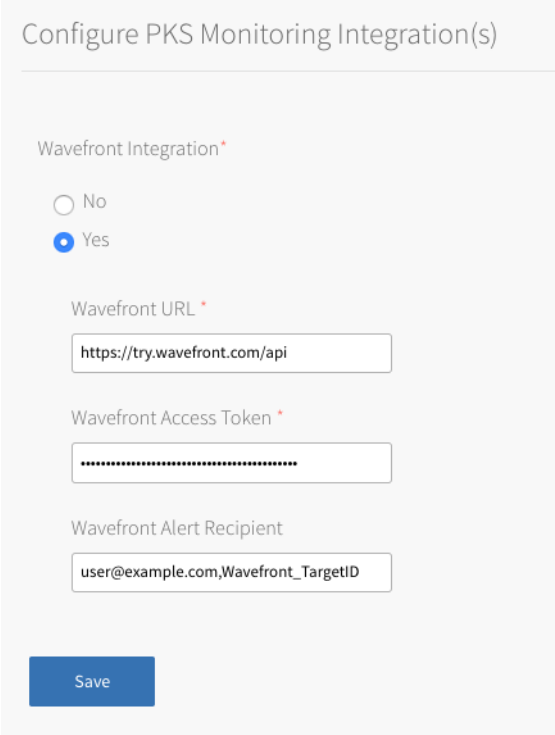
(Optional) Monitoring

You can monitor Kubernetes clusters and pods metrics externally using the integration with Wavefront by VMware.

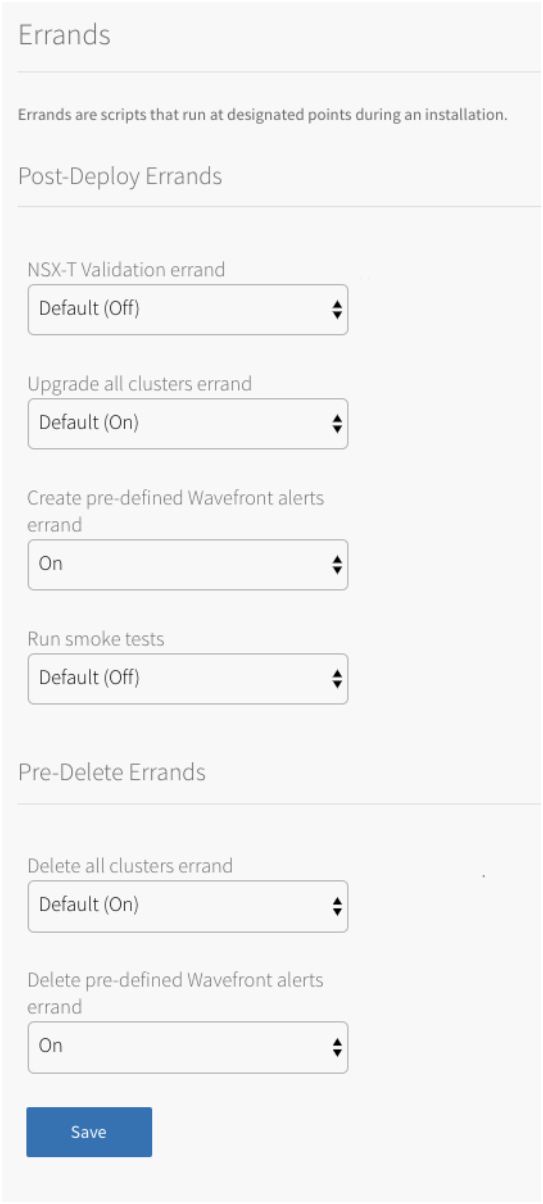
Note: Before you configure Wavefront integration, you must have an active Wavefront account and access to a Wavefront instance. You provide your Wavefront access token during configuration and enabling errands. For additional information, see Pivotal Container Service Integration Details in the Wavefront documentation.

By default, monitoring is disabled. To enable and configure Wavefront monitoring, do the following:

1. Select **Monitoring**.



2. On the **Monitoring** pane, under **Wavefront Integration**, select **Yes**.
3. Under **Wavefront URL**, enter the URL of your Wavefront subscription. For example, `https://try.wavefront.com/api`.
4. Under **Wavefront Access Token**, enter the API token for your Wavefront subscription.
5. To configure Wavefront to send alerts by email, enter email addresses or Wavefront Target IDs separated by commas under **Wavefront Alert Recipient**. For example, `user@example.com,Wavefront_TargetID`. To create alerts, you must enable errands.
6. Select **Errands**.
7. On the **Errands** pane, enable **Create pre-defined Wavefront alerts errand**.



8. Enable **Delete pre-defined Wavefront alerts errand**.
9. Click **Save**. Your settings apply to any clusters created after you have saved these configuration settings and clicked **Apply Changes**.

> **Note**: The PKS tile does not validate your Wavefront configuration settings. To verify your setup, look for cluster and pod metrics in Wavefront.

Usage Data

VMware's Customer Experience Improvement Program (CEIP) and the Pivotal Telemetry Program (Telemetry) provides VMware and Pivotal with information that enables the companies to improve their products and services, fix problems, and advise you on how best to deploy and use our products. As part of the CEIP and Telemetry, VMware and Pivotal collect technical information about your organization's use of the Pivotal Container Service (PKS) on a regular basis. Since PKS is jointly developed and sold by VMware and Pivotal, we will share this information with one another. Information collected under CEIP or Telemetry does not personally identify any individual.

Regardless of your selection in the **Usage Data** pane, a small amount of data is sent from Cloud Foundry Container Runtime (CFCR) to the PKS tile. However, that data is not shared externally.

To configure the **Usage Data** pane, perform the following steps:

1. Select the **Usage Data** side-tab.
2. Read the Usage Data description.

3. Make your selection.

    a. To join the program, select **Yes, I want to join the CEIP and Telemetry Program for PKS**.
    b. To decline joining the program, select **No, I do not want to join the CEIP and Telemetry Program for PKS**.

4. Click **Save**.

Note: If you join the CEIP and Telemetry Program for PKS, open your firewall to allow outgoing access to `https://vcsa.vmware.com/ph-prd` on port `443`.

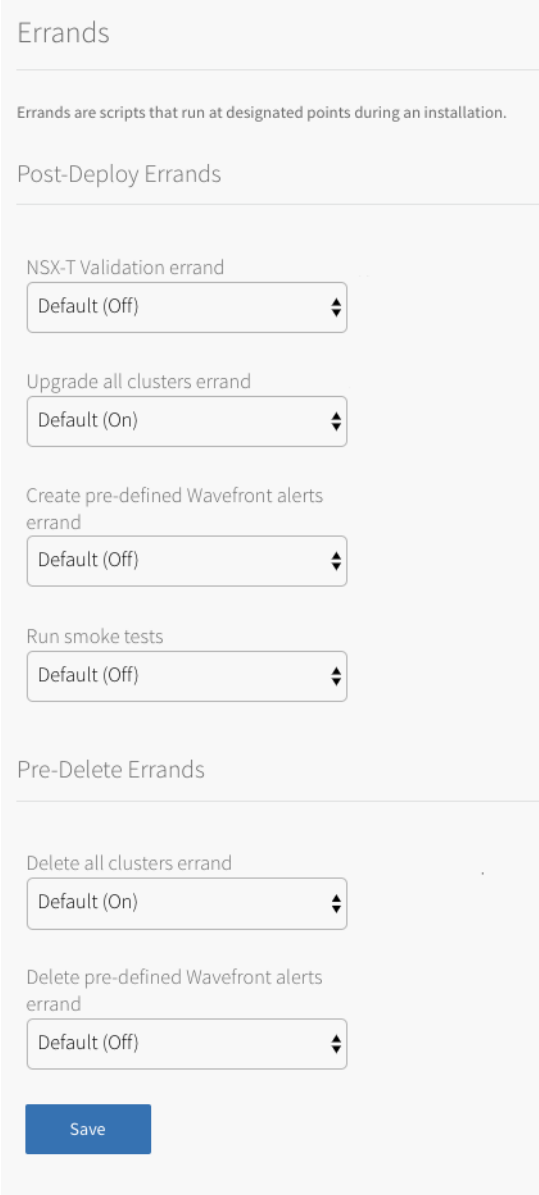
Errands

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

To configure when post-deploy and pre-delete errands for PKS are run, make a selection in the dropdown next to the errand.

We recommend that you set the **Run smoke tests** errand to **On**. The errand uses the PKS Command Line Interface (PKS CLI) to create a Kubernetes cluster and then delete it. If the creation or deletion fails, the errand fails and the installation of the PKS tile is aborted.

For the other errands, we recommend that you leave the default settings.



For more information about errands and their configuration state, see Managing Errands in Ops Manager.

WARNING: Because PKS uses floating stemcells, updating the PKS tile with a new stemcell triggers the rolling of every VM in each cluster. Also, updating other product tiles in your deployment with a new stemcell causes the PKS tile to roll VMs. This rolling is enabled by the **Upgrade all clusters errand**. We recommend that you keep this errand turned on because automatic rolling of VMs ensures that all deployed cluster VMs are patched. However, automatic rolling can cause downtime in your deployment.

If you are upgrading PKS, you must enable the **Upgrade All Clusters** errand.

Resource Config

VMs used by **Pivotal Container Service** jobs must meet the following minimum requirements:

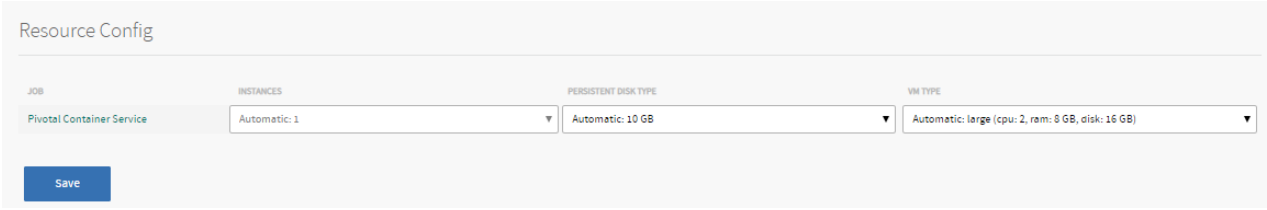
CPU	Memory	Disk
2	8 GB	29 GB

Note: If you experience timeouts or slowness when interacting with the PKS API, select a **VM Type** with greater CPU and memory resources.

To deploy **Pivotal Container Service** job VMs meeting the minimum requirements, perform the following steps:

1. Click **Resource Config**.

   The default "Automatic" **VM Type** does not meet the **Pivotal Container Service** job VM minimum requirements:

2. Select a **VM Type** with CPU, memory and disk resources either matching or exceeding the minimum **Pivotal Container Service** job VM requirements.

3. Select **Save**.

Step 3: Apply Changes

1. Return to the Ops Manager Installation Dashboard.

2. Click **Review Pending Changes**. Select the product that you intend to deploy and review the changes. For more information, see Reviewing Pending Product Changes.

3. Click **Apply Changes**.

Step 4: Retrieve the PKS API Endpoint

You must share the PKS API endpoint to allow your organization to use the API to create, update, and delete clusters. For more information, see Creating Clusters.

To retrieve the PKS API endpoint, do the following:

1. Navigate to the Ops Manager Installation Dashboard.

2. Click the Pivotal Container Service tile.

3. Click the **Status** tab and locate the **Pivotal Container Service** job. The IP address of the Pivotal Container Service job is the PKS API endpoint.

Step 5: Configure External Load Balancer

After you install the PKS tile, configure an external load balancer to access the PKS API from outside the network. You can use any external load balancer.

Your external load balancer forwards traffic to the PKS API endpoint on ports 8443 and 9021. Configure the external load balancer to resolve to the domain name you set in the PKS API section of the tile configuration.

Configure your load balancer with the following information:

- IP address from Retrieve PKS API Endpoint
- Ports 8443 and 9021
- HTTPS or TCP protocol

Step 6: Install the PKS and Kubernetes CLIs

The PKS and Kubernetes CLIs help you interact with your PKS-provisioned Kubernetes clusters and Kubernetes workloads. To install the CLIs, follow the instructions below:

- Installing the PKS CLI
- Installing the Kubernetes CLI

Step 7: Configure PKS API Access

Follow the procedures in Configuring PKS API Access.

Step 8: Configure Authentication for PKS

Configure authentication for PKS using User Account and Authentication (UAA). For information, see Managing Users in PKS with UAA.

Next Steps

After installing PKS on vSphere, you may want to do the following:

- Integrate VMware Harbor with PKS to store and manage container images. For more information, see Integrating VMware Harbor Registry with PKS.
- Create your first PKS cluster. For more information, see Creating Clusters.

Please send any feedback you have to pks-feedback@pivotal.io.

Installing PKS on vSphere with NSX-T Data Center

This topic lists the sections to follow when installing PKS on vSphere with NSX-T Data Center.

Preparing to Install PKS on vSphere with NSX-T

In preparation for installing PKS on vSphere with NSX-T, review the following documentation:

- Hardware Requirements for Deploying PKS on vSphere with NSX-T
- Firewall Ports and Protocols Requirements
- NSX-T Deployment Topologies for PKS
- vSphere with NSX-T Cluster Objects
- Preparing to Deploy PKS with NSX-T on vSphere

Installing PKS on vSphere with NSX-T

To install PKS on vSphere with NSX-T, complete the instructions in each of the following sections in the order listed:

- Deploying NSX-T v2.3.1 for PKS
- Deploying NSX-T v2.4.1 for PKS
- Deploying NSX-T v2.3.1 for PKS
- Deploying NSX-T v2.4.1 for PKS
- Creating the PKS Management Plane
- Creating the PKS Compute Plane
- Deploying Ops Manager with NSX-T for PKS
- Generating and Registering the NSX Manager Certificate for PKS
- Configuring BOSH Director with NSX-T for PKS
- Generating and Registering the NSX Manager Superuser Principal Identity Certificate and Key for PKS
- Creating NSX-T Objects for PKS
- Installing PKS on vSphere with NSX-T

Post-Installation NSX-T Configurations

After you have installed PKS on vSphere with NSX-T, refer to the following sections for additional NSX-T configuration options:

- Implementing a Multi-Foundation PKS Deployment
- Using Proxies with PKS on NSX-T
- Defining Network Profiles
- Configuring Multiple Tier-0 Routers for Tenant Isolation
- Configuring Ingress Resources and Load Balancer Services

Installing the PKS and Kubernetes CLIs

The PKS and Kubernetes CLIs help you interact with your PKS-provisioned Kubernetes clusters and Kubernetes workloads. To install the CLIs, follow the instructions below:

- Installing the PKS CLI
- Installing the Kubernetes CLI

Installing Harbor Registry

To install Harbor Registry for PKS, see Integrating VMware Harbor with PKS.

Please send any feedback you have to pks-feedback@pivotal.io.

vSphere with NSX-T Version Requirements

Page last updated:

This topic describes the version requirements for installing Pivotal Container Service (PKS) on vSphere with NSX-T integration.

For prerequisites and resource requirements for installing PKS on vSphere without NSX-T integration, see vSphere Prerequisites and Resource Requirements.

For hardware and resource requirements for deploying PKS on vSphere with NSX-T in production environments, see Hardware Requirements for PKS on vSphere with NSX-T.

PKS supports air-gapped deployments on vSphere with or without NSX-T integration.

You can also configure integration with the Harbor tile, an enterprise-class registry server for container images. For more information, see VMware Harbor Registry in the *Pivotal Partner documentation*.

vSphere Version Requirements

PKS on vSphere with NSX−T supports the following vSphere component versions:

Versions	Editions
• VMware vSphere 6.7 U1 EP06 (ESXi670-201901001) – for NSX-T 2.4 • VMware vSphere 6.7 U1 • VMware vSphere 6.7.0 • VMware vSphere 6.5 U2 P03 (ESXi650-201811002) – for NSX-T 2.4 • VMware vSphere 6.5 U2 • VMware vSphere 6.5 U1	• vSphere Enterprise Plus • vSphere with Operations Management Enterprise Plus

Note: VMware vSphere 6.7 is only supported with Ops Manager v2.3.1 or later and NSX−T v2.3.

For more information, see Upgrading vSphere in an NSX Environment in the VMware documentation.

NSX-T Integration Component Version Requirements

Refer to the PKS v1.3 Release Notes for supported NSX-T versions.

Hardware Requirements for PKS on vSphere with NSX-T

Page last updated:

This topic provides hardware requirements for production deployments of Pivotal Container Service (PKS) on vSphere with NSX-T.

vSphere Clusters for PKS

A vSphere cluster is a collection of ESXi hosts and associated virtual machines (VMs) with shared resources and a shared management interface. Installing PKS on vSphere with NSX-T requires the following vSphere clusters:

- PKS Management Cluster
- PKS Edge Cluster
- PKS Compute Cluster

For more information on creating vSphere clusters, see Creating Clusters in the vSphere documentation.

PKS Management Cluster

The PKS Management Cluster on vSphere comprises the following components:

- vCenter Server
- NSX-T Manager
- NSX-T Controller (quantity 3)

For more information, see Deploying NSX-T for PKS.

PKS Edge Cluster

The PKS Edge Cluster on vSphere comprises two or more NSX-T Edge Nodes in active/standby mode. The minimum number of Edge Nodes per Edge Cluster is two; the maximum is 10. PKS supports running Edge Node pairs in active/standby mode only.

For more information, see Deploying NSX-T for PKS.

PKS Compute Cluster

The PKS Compute Cluster on vSphere comprises the following components:

- Kubernetes master nodes (quantity 3)
- Kubernetes worker nodes

For more information, see Installing PKS on vSphere with NSX-T.

PKS Management Plane Placement Considerations

The PKS Management Plane comprises the following components:

- Pivotal Ops Manager
- Pivotal BOSH Director
- PKS Control Plane
- VMware Harbor Registry

Depending on your design choice, PKS management components can be deployed in the PKS Management Cluster on the standard vSphere network or in the PKS Compute Cluster on the NSX-T-defined virtual network. For more information, see NSX-T Deployment Topologies for PKS.

Configuration Requirements for vSphere Clusters for PKS

For each vSphere cluster defined for PKS, the following configurations are required to support production workloads:

- The vSphere Distributed Resource Scheduler (DRS) is enabled. For more information, see Creating a DRS Cluster in the vSphere documentation.
- The DRS custom automation level is set to **Partially Automated** or **Fully Automated**. For more information, see Set a Custom Automation Level for a Virtual Machine in the vSphere documentation.
- vSphere high-availability (HA) is enabled. For more information, see Creating and Using vSphere HA Clusters in the vSphere documentation.
- vSphere HA Admission Control (AC) is configured to support one ESXi host failure. For more information, see Configure Admission Control in the vSphere documentation.
  Specifically:
  - Host failure: Restart VMs
  - Admission Control: Host failures cluster tolerates = 1

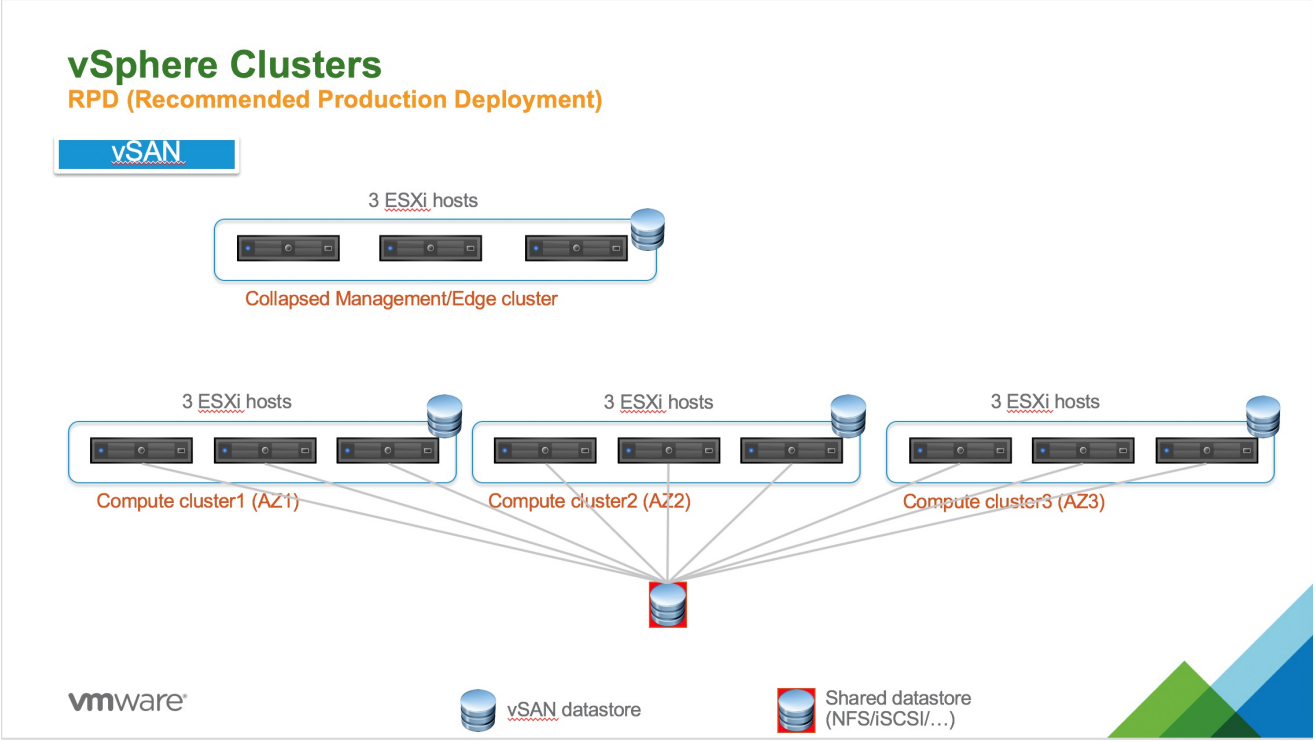
RPD for PKS on vSphere with NSX-T

The recommended production deployment (RPD) topology represents the VMware-recommended configuration to run production workloads in PKS on vSphere with NSX-T.

Note: The RPD differs depending on whether you are using vSAN or not.

RPD for PKS with vSAN

The RPD for PKS with vSAN storage requires 12 ESXi hosts. The diagram below shows the topology for this deployment.



The following subsections describe configuration details for the RPD with vSAN topology.

Management/Edge Cluster

The RPD with vSAN topology includes a Management/Edge Cluster with the following characteristics:

- Collapsed Management/Edge Cluster with three ESXi hosts.

- Each ESXi host runs one NSX-T Controller. The NSX-T Control Plane has three NSX-T Controllers total.
- Two NSX-T Edge Nodes are deployed across two different ESXi hosts.

Compute Clusters

The RPD with vSAN topology includes three Compute Clusters with the following characteristics:

- Each Compute cluster has three ESXi hosts and is bound by a distinct availability zone (AZ) defined in BOSH Director.
  - Compute cluster1 (AZ1) with three ESXi hosts.
  - Compute cluster2 (AZ2) with three ESXi hosts.
  - Compute cluster3 (AZ3) with three ESXi hosts.
- Each Compute cluster runs one instance of a PKS-provisioned Kubernetes cluster with three master nodes per cluster and a per-plan number of worker nodes.

Storage (vSAN)

The RPD with vSAN topology requires the following storage configuration:

- Each Compute Cluster is backed by a vSAN datastore
- An external shared datastore (using NFS or iSCSI, for instance) must be provided to store Kubernetes Pod PV (Persistent Volumes).
- Three ESXi hosts are required per Compute cluster because of the vSAN cluster requirements. For data protection, vSAN creates two copies of the data and requires one witness.

For more information on using vSAN with PKS, see PersistentVolume Storage Options on vSphere.

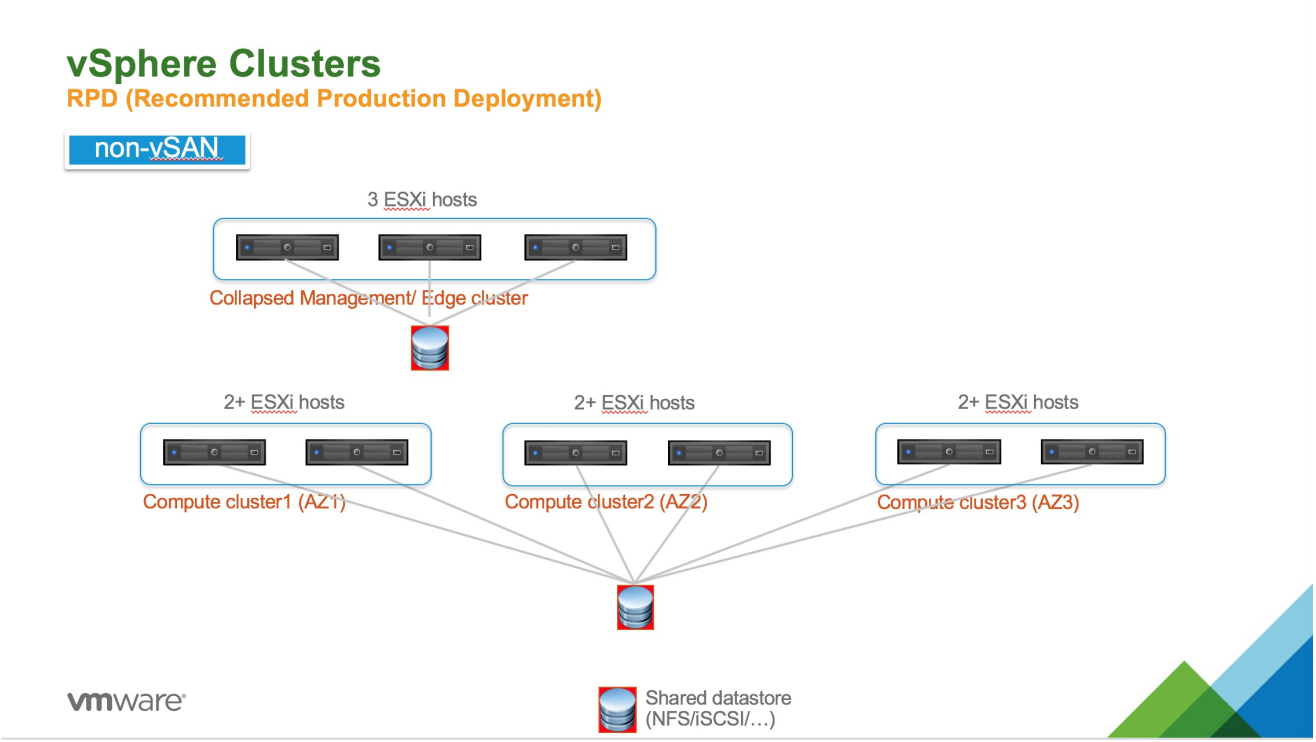
Future Growth

The RPD with vSAN topology can be scaled as follows to accommodate future growth requirements:

- The collapsed Management/Edge Cluster can be expanded to include up to 64 ESXi hosts.
- Each Compute Cluster can be expanded to include up to 64 ESXi hosts.

RPD for PKS without vSAN

The RPD for PKS without vSAN storage requires nine ESXi hosts. The diagram below shows the topology for this deployment.



The following subsections describe configuration details for the RPD of PKS without vSAN.

Management/Edge Cluster

The RPD without vSAN includes a Management/Edge Cluster with the following characteristics:

- Collapsed Management/Edge Cluster with three ESXi hosts.
- Each ESXi host runs one NSX-T Controller. The NSX-T Control Plane has three NSX-T Controllers total.
- Two NSX-T Edge Nodes are deployed across two different ESXi hosts.

Compute Clusters

The RPD without vSAN topology includes three Compute Clusters with the following characteristic:

- Each Compute cluster has two ESXi hosts and is bound by a distinct availability zone (AZ) defined in BOSH Director.
  - Compute cluster1 (AZ1) with two ESXi hosts.
  - Compute cluster2 (AZ2) with two ESXi hosts.
  - Compute cluster3 (AZ3) with two ESXi hosts.
- Each Compute cluster runs one instance of a PKS-provisioned Kubernetes cluster with three master nodes per cluster and a per-plan number of worker nodes.

Storage (non-vSAN)

The RPD without vSAN topology requires the following storage configuration:

- All Compute Clusters are connected to same shared datastore that is used for persistent VM disks for PKS components and Persistent Volumes (PVs) for Kubernetes pods.
- All datastores can be collapses to single datastore, if needed.

Future Growth

The RPD without vSAN topology can be scaled as follows to accommodate future growth requirements:

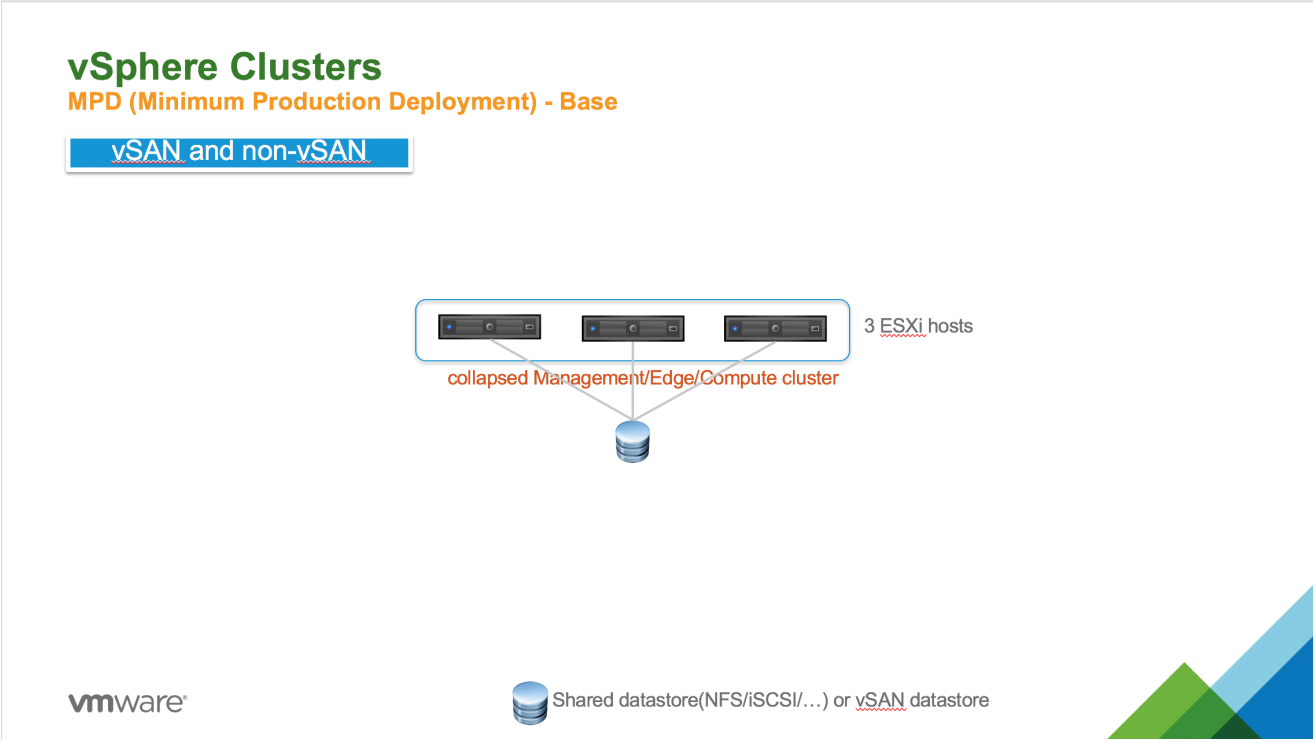
- The collapsed Management/Edge Cluster can be expanded to include up to 64 ESXi hosts.
- Each Compute Cluster can be expanded to include up to 64 ESXi hosts.

MPD for PKS on vSphere with NSX-T

The minimum production deployment (MPD) topology represents the baseline requirements for running PKS on vSphere with NSX-T.

Note: The MPD topology for PKS applies to both vSAN and non-vSAN environments.

The diagram below shows the topology for this deployment.



The following subsections describe configuration details for an MPD of PKS.

MPD Topology

The MPD topology for PKS requires the following minimum configuration:

- A single collapsed Management/Edge/Compute cluster running three ESXi hosts in total.
- Each ESXi host runs one NSX-T Controller. The NSX-T Control Plane has three NSX-T Controllers in total.
- Each ESXi host runs one Kubernetes master node. Each Kubernetes cluster has three master nodes in total.
- Two NSX-T edge nodes are deployed across two different ESXi hosts.
- The shared datastore (NFS or iSCSI, for instance) or vSAN datastore is used for persistent VM disks for PKS components and Persistent Volumes (PVs) for Kubernetes pods.
- The collapsed Management/Edge/Compute cluster can be expanded to include up to 64 ESXi hosts.

Note: For an MPD deployment, each ESXi host must have four physical network interface controllers (PNICs). In addition, while a PKS deployment requires a minimum of three nodes, PKS upgrades require four ESXi hosts to ensure full survivability of the NSX Manager appliance.

MPD Configuration Requirements

When configuring vSphere for an MPD topology for PKS, keep in mind the following requirements:

- When deploying the NSX-T Controller to each ESXi host, create a vSphere distributed resource scheduler (DRS) anti-affinity rule of type “separate virtual machines” for each of the three NSX-T Controllers.
- When deploying the NSX-T Edge Nodes across two different ESXi hosts, create a DRS anti-affinity rule of type “separate virtual machines” for both Edge Node VMs.
- After deploying the Kubernetes cluster, you must manually make sure each master node is deployed to a different ESXi host by tuning the DRS anti-affinity rule of type “separate virtual machines.”

For more information on defining DRS anti-affinity rules, see Virtual Machine Storage DRS Rules in the vSphere documentation.

MPD Considerations

When planning an MPD topology for PKS, keep in mind the following:

- Leverage vSphere resource pools to allocate proper hardware resources for the PKS Management Plane components and tune reservation and resource limits accordingly.
- There is no fault tolerance for the Kubernetes cluster because PKS Availability Zones are not fully leveraged with this topology.
- At a minimum, the PKS AZ should be mapped to a vSphere Resource Pool.

For more information, see Creating the PKS Management Plane and Creating the PKS Compute Plane.

VM Inventory and Sizes

The following tables list the VMs and their sizes for deployments of PKS on vSphere with NSX-T.

Control Plane VMs and Sizes

The following table lists the resource requirements for each VM in the PKS infrastructure and control plane.

VM	CPU	Memory (GB)	Disk Space (GB)
vCenter Appliance	4	16	290
NSX-T Manager	4	16	140
NSX-T Controller 1	4	16	120
NSX-T Controller 2	4	16	120
NSX-T Controller 3	4	16	120
Ops Manager	1	8	160
BOSH Director	2	8	103
PKS Control Plane	2	8	29 ^*
Harbor Registry	2	8	167
TOTAL	**27**	**112**	**1.25 TB**

Storage Requirements for Large Numbers of Pods

If you expect the cluster workload to run a large number of pods continuously, then increase the size of persistent disk storage allocated to the the Pivotal Container Service VM as follows:

Number of Pods	Storage (Persistent Disk) Requirement ^*
1,000 pods	20 GB
5,000 pods	100 GB
10,000 pods	200 GB
50,000 pods	1,000 GB

NSX-T Edge Node VMs and Sizes

The following table lists the resource requirements for each VM in the Edge Cluster.

VM	CPU (Intel CPU only)	Memory (GB)	Disk Space (GB)
NSX-T Edge Node 1	8	16	120
NSX-T Edge Node 2	8	16	120
TOTAL	**16**	**32**	**.25 TB**

Note: NSX-T Edge Nodes must be deployed on Intel-based hardware.

Kubernetes Cluster Nodes VMs and Sizes

The following table lists sizing information for Kubernetes cluster node VMs. The size and resource consumption of these VMs are configurable in the **Plans** section of the PKS tile.

VM	CPU	Memory (GB)	Ephemeral Disk Space	Persistent Disk Space
Master Nodes	1 to 16	1 to 64	8 to 256 GB	1 GB to 32 TB
Worker Nodes	1 to 16	1 to 64	8 to 256 GB	1 GB to 32 TB

For illustrative purposes, the following table shows sizing information for two example Kubernetes clusters. Each cluster has three master nodes and five worker nodes.

VM	CPU per Node	Memory (GB) per Node	Ephemeral Disk Space per Node	Persistent Disk Space per Node
Master Nodes (6 total)	2	8	64 GB	128 GB
Worker Nodes (10 total)	4	16	64 GB	256 GB
TOTAL	**52**	**208**	**1.0 TB**	**3.4 TB**

Hardware Requirements

The following tables list the hardware requirements for RDP and MPD topologies for PKS on vSphere with NSX-T.

RPD Hardware Requirements

The following table lists the hardware requirements for the RPD with vSAN topology.

VM	Number of Hosts	Total Cores per Host (with HT)	Memory per Host (GB)	NICs per Host	Shared Datastore
Management/Edge Cluster	3	16	98	2x 10GbE	1.5 TB
Compute cluster1 (AZ1)	3	6	48	2x 10GbE	192 GB
Compute cluster2 (AZ2)	3	6	48	2x 10GbE	192 GB
Compute cluster3 (AZ3)	3	6	48	2x 10GbE	192 GB

Note: The Total Cores per Host values assume the use of hyper-threading (HT).

The following table lists the hardware requirements for the RPD without vSAN topology.

VM	Number of Hosts	Total Cores per Host (with HT)	Memory per Host (GB)	NICs per Host	Shared Datastore
Management/Edge Cluster	3	16	98	2x 10GbE	1.5 TB
Compute cluster1 (AZ1)	2	10	70	2x 10GbE	192 GB
Compute cluster2 (AZ2)	2	10	70	2x 10GbE	192 GB
Compute cluster3 (AZ3)	2	10	70	2x 10GbE	192 GB

Note: The Total Cores per Host values assume the use of hyper-threading (HT).

MPD Hardware Requirements

The following table lists the hardware requirements for the MPD topology with a single (collapsed) cluster for all Management, Edge, and Compute nodes.

VM	Number of Hosts	Total Cores per Host	Memory per Host (GB)	NICs per Host	Shared Datastore
Collapsed Cluster	3	32 (with hyper-threading)	236	2x 10GbE	5.9 TB

Adding Hardware Capacity

To add hardware capacity to your PKS environment on vSphere, do the following: 1. Add one or more ESXi hosts to the vSphere compute cluster. For more information, see the VMware vSphere documentation. 1. Prepare each newly added ESXi host so that it becomes an ESXi transport node for NSX-T. For more information, see Prepare ESXi Servers for the PKS Compute Cluster.

Please send any feedback you have to pks-feedback@pivotal.io.

Firewall Ports and Protocols Requirements

Page last updated:

This topic describes the firewall ports and protocols requirements for using Pivotal Container Service (PKS) on vSphere with NSX-T integration.

Firewalls and security policies are used to filter traffic and limit access in environments with strict inter-network access control policies.

Apps frequently require the ability to pass internal communication between system components on different networks and require one or more conduits through the environment's firewalls. Firewall rules are also required to enable interfacing with external systems such as with enterprise apps or apps and data on the public Internet.

For PKS, Pivotal recommends that you disable security policies that filter traffic between the networks supporting the system. To secure the environment and grant access between system components with PKS, use one of the following methods:

- Enable access to apps through standard Kubernetes load-balancers and ingress controller types. This enables you to designate specific ports and protocols as a firewall conduit.
- Enable access using the NSX-T load balancer and ingress. This enables you to configure external addresses and ports that are automatically mapped and resolved to internal/local addresses and ports.

For information on ports and protocol requirements for vSphere without NSX-T, see Firewall Ports and Protocols Requirements for vSphere without NSX-T

If you are unable to implement your security policy using the methods described above, refer to the following table, which identifies the flows between system components in a typical PKS deployment.

Note: To control which groups access deploying and scaling your organization's Enterprise PKS-deployed Kubernetes clusters, configure your firewall settings as described on the Operator –> PKS API server lines below.

PKS Ports and Protocols

The following tables list ports and protocols required for network communications between PKS v1.3.6 and later, and vSphere 6.7 and NSX-T 2.4.0.1 and later.

PKS Users Ports and Protocols

The following table lists ports and protocols used for network communication between PKS user interface components.

Source Component	Destination Component	Destination Protocol	Destination Port	Service
Admin/Operator Console	All System Components	TCP	22	ssh
Admin/Operator Console	All System Components	TCP	80	http
Admin/Operator Console	All System Components	TCP	443	https
Admin/Operator Console	Cloud Foundry BOSH Director	TCP	25555	bosh director rest api
Admin/Operator Console	NSX-T API VIP	TCP	443	https
Admin/Operator Console	Pivotal Cloud Foundry Operations Manager	TCP	22	ssh
Admin/Operator Console	Pivotal Cloud Foundry Operations Manager	TCP	443	https
Admin/Operator Console	PKS Controller	TCP	9021	pks api server
Admin/Operator Console	vCenter Server	TCP	443	https
Admin/Operator Console	vCenter Server	TCP	5480	vami
Admin/Operator Console	vSphere ESXI Hosts Mgmt. vmknic	TCP	902	ideafarm-door
Admin/Operator and Developer Consoles	Harbor Private Image Registry	TCP	80	http
Admin/Operator and Developer Consoles	Harbor Private Image Registry	TCP	443	https

Source Component	Destination Component	Destination Protocol	Destination Port	Service
Admin/Operator and Developer Consoles	Harbor Private Image Registry	TCP	4443	notary
Admin/Operator and Developer Consoles	Kubernetes App Load-Balancer Svc	TCP/UDP	Varies	varies with apps
Admin/Operator and Developer Consoles	Kubernetes Cluster API Server -LB VIP	TCP	8443	httpsca
Admin/Operator and Developer Consoles	Kubernetes Cluster Ingress Controller	TCP	80	http
Admin/Operator and Developer Consoles	Kubernetes Cluster Ingress Controller	TCP	443	https
Admin/Operator and Developer Consoles	Kubernetes Cluster Worker Node	TCP/UDP	30000-32767	kubernetes nodeport
Admin/Operator and Developer Consoles	PKS Controller	TCP	8443	httpsca
All User Consoles (Operator, Developer, Consumer)	Kubernetes App Load-Balancer Svc	TCP/UDP	Varies	varies with apps
All User Consoles (Operator, Developer, Consumer)	Kubernetes Cluster Ingress Controller	TCP	80	http
All User Consoles (Operator, Developer, Consumer)	Kubernetes Cluster Ingress Controller	TCP	443	https
All User Consoles (Operator, Developer, Consumer)	Kubernetes Cluster Worker Node	TCP/UDP	30000-32767	kubernetes nodeport

Note: The `type:NodePort` Service type is not supported for PKS deployments on vSphere with NSX-T. Only `type:LoadBalancer` and Services associated with Ingress rules are supported on vSphere with NSX-T.

PKS Core Ports and Protocols

The following table lists ports and protocols used for network communication between core PKS components.

Source Component	Destination Component	Destination Protocol	Destination Port	Service
All System Components	Corporate Domain Name Server	TCP/UDP	53	dns
All System Components	Network Time Server	UDP	123	ntp
All System Components	vRealize LogInsight	TCP/UDP	514/1514	syslog/tls syslog
All System Control Plane Components	AD/LDAP Directory Server	TCP/UDP	389/636	ldap/ldaps
Pivotal Cloud Foundry Operations Manager	Admin/Operator Console	TCP	22	ssh
Pivotal Cloud Foundry Operations Manager	Cloud Foundry BOSH Director	TCP	6868	bosh agent http
Pivotal Cloud Foundry Operations Manager	Cloud Foundry BOSH Director	TCP	8443	httpsca
Pivotal Cloud Foundry Operations Manager	Cloud Foundry BOSH Director	TCP	8844	credhub
Pivotal Cloud Foundry Operations Manager	Cloud Foundry BOSH Director	TCP	25555	bosh director rest api
Pivotal Cloud Foundry Operations Manager	Harbor Private Image Registry	TCP	22	ssh
Pivotal Cloud Foundry Operations Manager	Kubernetes Cluster Master/Etcd Node	TCP	22	ssh
Pivotal Cloud Foundry Operations Manager	Kubernetes Cluster Worker Node	TCP	22	ssh
Pivotal Cloud Foundry Operations Manager	NSX-T API VIP	TCP	443	https
Pivotal Cloud Foundry Operations Manager	NSX-T Manager/Controller Node	TCP	22	ssh
Pivotal Cloud Foundry Operations Manager	NSX-T Manager/Controller Node	TCP	443	https
Pivotal Cloud Foundry Operations				

Source Component	Destination Component	Destination Protocol	Destination Port	Service
Manager	PKS Controller	TCP	22	ssh
Pivotal Cloud Foundry Operations Manager	PKS Controller	TCP	8443	httpsca
Pivotal Cloud Foundry Operations Manager	vCenter Server	TCP	443	https
Pivotal Cloud Foundry Operations Manager	vSphere ESXI Hosts Mgmt. vmknic	TCP	443	https
Cloud Foundry BOSH Director	NSX-T API VIP	TCP	443	https
Cloud Foundry BOSH Director	vCenter Server	TCP	443	https
Cloud Foundry BOSH Director	vSphere ESXI Hosts Mgmt. vmknic	TCP	443	https
BOSH Compilation Job VM	Cloud Foundry BOSH Director	TCP	4222	bosh nats server
BOSH Compilation Job VM	Cloud Foundry BOSH Director	TCP	25250	bosh blobstore
BOSH Compilation Job VM	Cloud Foundry BOSH Director	TCP	25923	health monitor daemon
BOSH Compilation Job VM	Harbor Private Image Registry	TCP	443	https
BOSH Compilation Job VM	Harbor Private Image Registry	TCP	8853	bosh dns health
PKS Controller	Cloud Foundry BOSH Director	TCP	4222	bosh nats server
PKS Controller	Cloud Foundry BOSH Director	TCP	8443	httpsca
PKS Controller	Cloud Foundry BOSH Director	TCP	25250	bosh blobstore
PKS Controller	Cloud Foundry BOSH Director	TCP	25555	bosh director rest api
PKS Controller	Cloud Foundry BOSH Director	TCP	25923	health monitor daemon
PKS Controller	Kubernetes Cluster Master/Etcd Node	TCP	8443	httpsca
PKS Controller	NSX-T API VIP	TCP	443	https
PKS Controller	vCenter Server	TCP	443	https
Harbor Private Image Registry	Cloud Foundry BOSH Director	TCP	4222	bosh nats server
Harbor Private Image Registry	Cloud Foundry BOSH Director	TCP	25250	bosh blobstore
Harbor Private Image Registry	Cloud Foundry BOSH Director	TCP	25923	health monitor daemon
Harbor Private Image Registry	IP NAS Storage Array	TCP	111	nfs rpc portmapper
Harbor Private Image Registry	IP NAS Storage Array	TCP	2049	nfs
Harbor Private Image Registry	Public CVE Source Database	TCP	443	https
kube-system pod/telemetry-agent	PKS Controller	TCP	24224	fluentd out_forward
Kubernetes Cluster Ingress Controller	NSX-T API VIP	TCP	443	https
Kubernetes Cluster Ingress Controller	vCenter Server	TCP	443	https
Kubernetes Cluster Master/Etcd Node	Cloud Foundry BOSH Director	TCP	4222	bosh nats server
Kubernetes Cluster Master/Etcd Node	Cloud Foundry BOSH Director	TCP	25250	bosh blobstore
Kubernetes Cluster Master/Etcd Node	Cloud Foundry BOSH Director	TCP	25923	health monitor daemon
Kubernetes Cluster Master/Etcd Node	Kubernetes Cluster Master/Etcd Node	TCP	2379	etcd clent
Kubernetes Cluster Master/Etcd Node	Kubernetes Cluster Master/Etcd Node	TCP	2380	etcd server
Kubernetes Cluster Master/Etcd Node	Kubernetes Cluster Master/Etcd Node	TCP	8443	httpsca
Kubernetes Cluster Master/Etcd Node	Kubernetes Cluster Master/Etcd Node	TCP	8853	bosh dns health
Kubernetes Cluster Master/Etcd Node	Kubernetes Cluster Worker Node	TCP	4194	cadvisor
Kubernetes Cluster Master/Etcd Node	Kubernetes Cluster Worker Node	TCP	10250	kubelet api
Kubernetes Cluster Master/Etcd Node	Kubernetes Cluster Worker Node	TCP	31194	cadvisor

Source Component	Destination Component	Destination Protocol	Destination Port	Service
Kubernetes Cluster Master/Etcd Node	NSX-T API VIP	TCP	443	https
Kubernetes Cluster Master/Etcd Node	PKS Controller	TCP	8443	httpsca
Kubernetes Cluster Master/Etcd Node	PKS Controller	TCP	8853	bosh dns health
Kubernetes Cluster Master/Etcd Node	vCenter Server	TCP	443	https
Kubernetes Cluster Worker Node	Cloud Foundry BOSH Director	TCP	4222	bosh nats server
Kubernetes Cluster Worker Node	Cloud Foundry BOSH Director	TCP	25250	bosh blobstore
Kubernetes Cluster Worker Node	Cloud Foundry BOSH Director	TCP	25923	health monitor daemon
Kubernetes Cluster Worker Node	Harbor Private Image Registry	TCP	443	https
Kubernetes Cluster Worker Node	Harbor Private Image Registry	TCP	8853	bosh dns health
Kubernetes Cluster Worker Node	IP NAS Storage Array	TCP	111	nfs rpc portmapper
Kubernetes Cluster Worker Node	IP NAS Storage Array	TCP	2049	nfs
Kubernetes Cluster Worker Node	Kubernetes Cluster Master/Etcd Node	TCP	8443	httpsca
Kubernetes Cluster Worker Node	Kubernetes Cluster Master/Etcd Node	TCP	8853	bosh dns health
Kubernetes Cluster Worker Node	Kubernetes Cluster Master/Etcd Node	TCP	10250	kubelet api
pks-system pod/cert-generator	PKS Controller	TCP	24224	fluentd out_forward
pks-system pod/fluent-bit	PKS Controller	TCP	24224	fluentd out_forward

VMWare Ports and Protocols

The following tables list ports and protocols required for network communication between VMware components.

VMWare Virtual Infrastructure Ports and Protocols

The following table lists ports and protocols used for network communication between VMware virtual infrastructure components.

Source Component	Destination Component	Destination Protocol	Destination Port	Service
vCenter Server	NSX-T Manager/Controller Node	TCP	8080	http alt
vCenter Server	vSphere ESXI Hosts Mgmt. vmknic	TCP	443	https
vCenter Server	vSphere ESXI Hosts Mgmt. vmknic	TCP	8080	http alt
vCenter Server	vSphere ESXI Hosts Mgmt. vmknic	TCP	9080	io filter storage
vSphere ESXI Hosts Mgmt. vmknic	NSX-T Manager/Controller Node	TCP	443	https
vSphere ESXI Hosts Mgmt. vmknic	NSX-T Manager/Controller Node	TCP	1235	netcpa
vSphere ESXI Hosts Mgmt. vmknic	NSX-T Manager/Controller Node	TCP	5671	amqp traffic
vSphere ESXI Hosts Mgmt. vmknic	NSX-T Manager/Controller Node	TCP	8080	http alt
vSphere ESXI Hosts Mgmt. vmknic	vCenter Server	UDP	902	ideafarm-door
vSphere ESXI Hosts Mgmt. vmknic	vCenter Server	TCP	9084	update manager
vSphere ESXI Hosts Mgmt. vmknic	vSphere ESXI Hosts Mgmt. vmknic	TCP	8182	vsphere ha
vSphere ESXI Hosts Mgmt. vmknic	vSphere ESXI Hosts Mgmt. vmknic	UDP	8182	vsphere ha
vSphere ESXI Hosts vMotion vmknic	vSphere ESXI Hosts vMotion vmknic	TCP	8000	vmotion
vSphere ESXI Hosts IP Storage vmknic	IP NAS Storage Array	TCP	111	nfs rpc portmapper
vSphere ESXI Hosts IP Storage vmknic	IP NAS Storage Array	TCP	2049	nfs
vSphere ESXI Hosts IP Storage vmknic	IP NAS Storage Array	TCP	3260	iscsi
vSphere ESXI Hosts vSAN vmknic	vSphere ESXI Hosts vSAN vmknic	TCP	2233	vsan transport
vSphere ESXI Hosts vSAN vmknic	vSphere ESXI Hosts vSAN vmknic	UDP	12321	unicast agent

Source Component	Destination Component	Destination Protocol	Destination Port	Service
vSphere ESXI Hosts vSAN vmknic	vSphere ESXI Hosts vSAN vmknic	UDP	12345	vsan cluster svc
vSphere ESXI Hosts vSAN vmknic	vSphere ESXI Hosts vSAN vmknic	UDP	23451	vsan cluster svc
vSphere ESXI Hosts TEP vmknic	vSphere ESXI Hosts TEP vmknic	UDP	3784	bfd
vSphere ESXI Hosts TEP vmknic	vSphere ESXI Hosts TEP vmknic	UDP	3785	bfd
vSphere ESXI Hosts TEP vmknic	vSphere ESXI Hosts TEP vmknic	UDP	6081	geneve
vSphere ESXI Hosts TEP vmknic	NSX-T Edge TEP vNIC	UDP	3784	bfd
vSphere ESXI Hosts TEP vmknic	NSX-T Edge TEP vNIC	UDP	3785	bfd
vSphere ESXI Hosts TEP vmknic	NSX-T Edge TEP vNIC	UDP	6081	geneve
NSX-T Manager/Controller Node	NSX-T API VIP	TCP	443	https
NSX-T Manager/Controller Node	NSX-T Manager/Controller Node	TCP	443	https
NSX-T Manager/Controller Node	NSX-T Manager/Controller Node	TCP	5671	amqp traffic
NSX-T Manager/Controller Node	NSX-T Manager/Controller Node	TCP	8080	http alt
NSX-T Manager/Controller Node	NSX-T Manager/Controller Node	TCP	9000	loginsight ingestion api
NSX-T Manager/Controller Node	Traceroute Destination	UDP	33434-33523	traceroute
NSX-T Manager/Controller Node	vCenter Server	TCP	80	http
NSX-T Manager/Controller Node	vCenter Server	TCP	443	https
NSX-T Manager/Controller Node	vSphere ESXI Hosts Mgmt. vmknic	TCP	443	https
NSX-T Edge Management	NSX-T Edge Management	TCP	1167	DHCP backend
NSX-T Edge Management	NSX-T Edge Management	TCP	2480	Nestdb
NSX-T Edge Management	NSX-T Edge Management	UDP	3784	bfd
NSX-T Edge Management	NSX-T Edge Management	UDP	50263	high-availability
NSX-T Edge Management	NSX-T Manager/Controller Node	TCP	443	https
NSX-T Edge Management	NSX-T Manager/Controller Node	TCP	1235	netcpa
NSX-T Edge Management	NSX-T Manager/Controller Node	TCP	5671	amqp traffic
NSX-T Edge Management	NSX-T Manager/Controller Node	TCP	8080	http alt
NSX-T Edge Management	Traceroute Destination	UDP	33434-33523	traceroute
NSX-T Edge TEP vNIC	NSX-T Edge TEP vNIC	UDP	3784	bfd
NSX-T Edge TEP vNIC	NSX-T Edge TEP vNIC	UDP	3785	bfd
NSX-T Edge TEP vNIC	NSX-T Edge TEP vNIC	UDP	6081	geneve
NSX-T Edge TEP vNIC	NSX-T Edge TEP vNIC	UDP	50263	high-availability
NSX-T Edge TEP vNIC	vSphere ESXI Hosts TEP vmknic	UDP	3784	bfd
NSX-T Edge TEP vNIC	vSphere ESXI Hosts TEP vmknic	UDP	3785	bfd
NSX-T Edge TEP vNIC	vSphere ESXI Hosts TEP vmknic	UDP	6081	geneve
NSX-T Edge Tier-0 Uplink IP(s) / HA VIP	Physical Network Router	TCP	179	bgp
Physical Network Router	NSX-T Edge Tier-0 Uplink IP(s) / HA VIP	TCP	179	bgp

VMWare Optional Integration Ports and Protocols

The following table lists ports and protocols used for network communication between optional VMware integrations.

Source Component	Destination Component	Destination Protocol	Destination Port	Service
Admin/Operator Console	vRealize Operations Manager	TCP	443	https
vRealize Operations Manager	Kubernetes Cluster API Server -LB VIP	TCP	8443	httpsca
vRealize Operations Manager	NSX-T API VIP	TCP	443	https
vRealize Operations Manager	PKS Controller	TCP	8443	httpsca
vRealize Operations Manager	Kubernetes Cluster API Server -LB VIP	TCP	8443	httpsca
Admin/Operator Console	vRealize LogInsight	TCP	443	https
Kubernetes Cluster Ingress Controller	vRealize LogInsight	TCP	9000	ingestion api

Source Component	Destination Component	Destination Protocol	Destination Port	Service
Kubernetes Cluster Master/Etcd Node	vRealize LogInsight	TCP	9000	ingestion api
Kubernetes Cluster Master/Etcd Node	vRealize LogInsight	TCP	9543	ingestion api -tls
Kubernetes Cluster Worker Node	vRealize LogInsight	TCP	9000	ingestion api
Kubernetes Cluster Worker Node	vRealize LogInsight	TCP	9543	ingestion api -tls
NSX-T Manager/Controller Node	vRealize LogInsight	TCP	9000	ingestion api
PKS Controller	vRealize LogInsight	TCP	9000	ingestion api
Admin/Operator and Developer Consoles	Wavefront SaaS APM	TCP	443	https
kube-system pod/wavefront-proxy	Wavefront SaaS APM	TCP	443	https
kube-system pod/wavefront-proxy	Wavefront SaaS APM	TCP	8443	httpsca
pks-system pod/wavefront-collector	PKS Controller	TCP	24224	fluentd out_forward
Admin/Operator Console	vRealize Network Insight Platform	TCP	443	https
Admin/Operator Console	vRealize Network Insight Proxy	TCP	22	ssh
vRealize Network Insight Proxy	Kubernetes Cluster API Server -LB VIP	TCP	8443	httpsca
vRealize Network Insight Proxy	NSX-T API VIP	TCP	22	ssh
vRealize Network Insight Proxy	NSX-T API VIP	TCP	443	https
vRealize Network Insight Proxy	PKS Controller	TCP	8443	httpsca
vRealize Network Insight Proxy	PKS Controller	TCP	9021	pks api server

Please send any feedback you have to pks-feedback@pivotal.io.

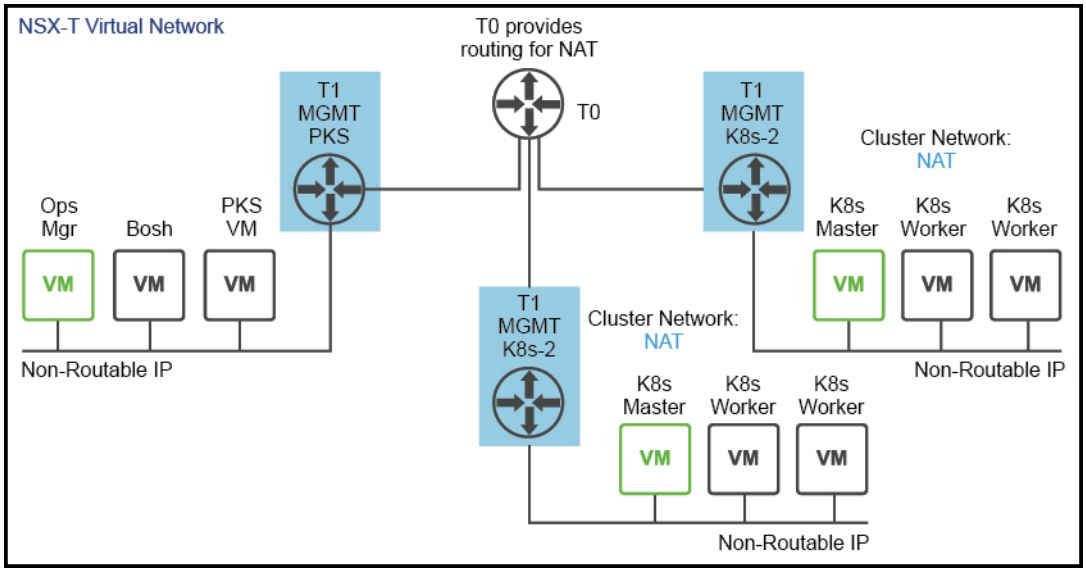
NSX-T Deployment Topologies for PKS

Page last updated:

There are three supported topologies in which to deploy NSX-T with PKS.

NAT Topology

The following figure shows a Network Address Translation (NAT) deployment:



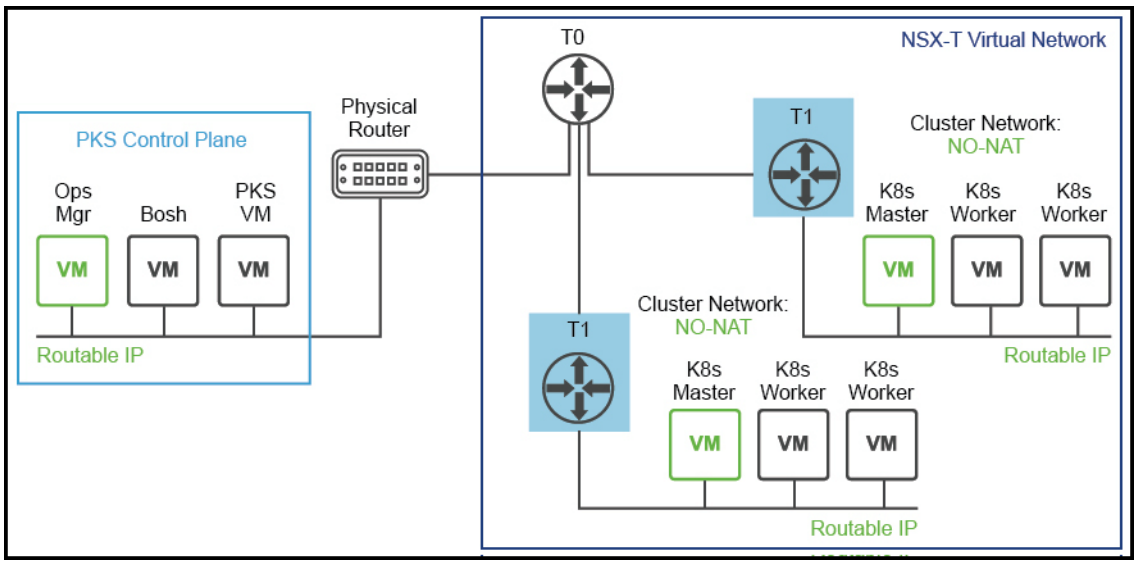
View a larger version of this image.

This topology has the following characteristics:

- PKS control plane (Ops Manager, BOSH Director, and PKS VM) components are all located on a logical switch that has undergone Network Address Translation on a T0.
- Kubernetes cluster master and worker nodes are located on a logical switch that has undergone Network Address Translation on a T0. This requires DNAT rules to allow access to Kubernetes APIs.

No-NAT with Virtual Switch (VSS/VDS) Topology

The following figure shows a No-NAT with Virtual Switch (VSS/VDS) deployment:



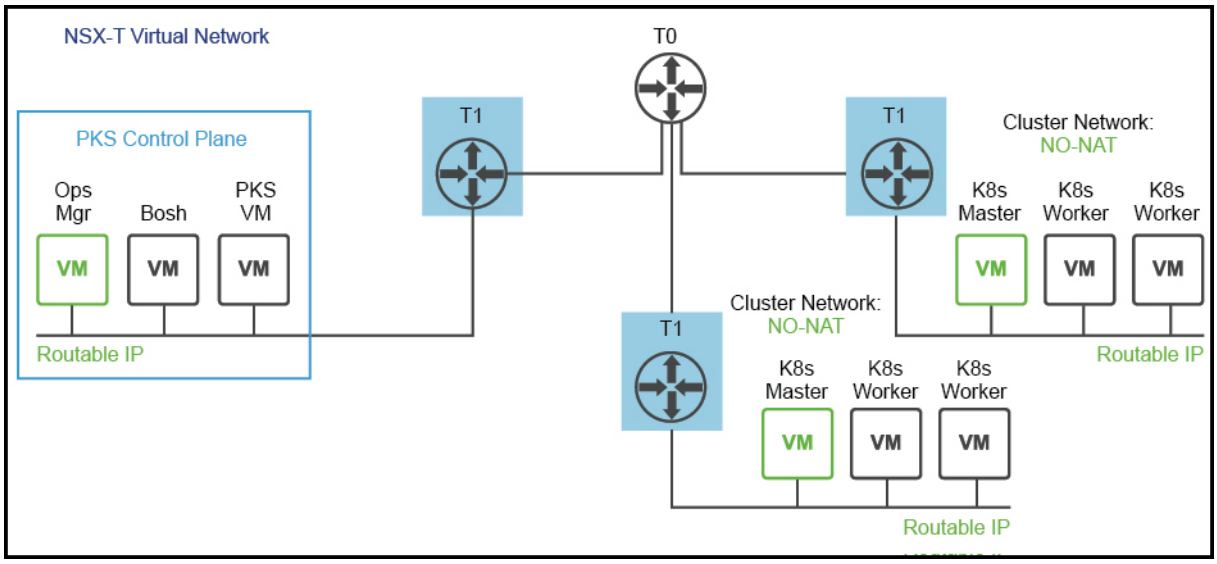
View a larger version of this image.

This topology has the following characteristics:

- PKS control plane (Ops Manager, BOSH Director, and PKS VM) components are using corporate routable IP addresses.
- Kubernetes cluster master and worker nodes are using corporate routable IP addresses.
- The PKS control plane is deployed outside of the NSX-T network and the Kubernetes clusters are deployed and managed within the NSX-T network. Since BOSH needs routable access to the Kubernetes Nodes to monitor and manage them, the Kubernetes Nodes need routable access.

No-NAT with Logical Switch (NSX-T) Topology

The following figure shows a No-NAT with Logical Switch (NSX-T) deployment:



View a larger version of this image.

This topology has the following characteristics:

- PKS control plane (Ops Manager, BOSH Director, and PKS VM) components are using corporate routable IP addresses.
- Kubernetes cluster master and worker nodes are using corporate routable IP addresses.
- The PKS control plane is deployed inside of the NSX-T network. Both the PKS control plane components (VMs) and the Kubernetes Nodes use corporate routable IP addresses.

vSphere with NSX-T Cluster Objects

Page last updated:

This topic lists and describes the vSphere VMs and NSX-T objects that Pivotal Container Service (PKS) creates when you create a Kubernetes cluster. When you delete a Kubernetes cluster, PKS removes these objects.

For information about creating a Kubernetes cluster using PKS, see Creating Clusters. For information about deleting a Kubernetes cluster using PKS, see Deleting Clusters.

vSphere Virtual Machines

When a new Kubernetes cluster is created, PKS creates the following virtual machines (VMs) in the designated vSphere cluster:

Object Number	Object Description
1 or 3	Kubernetes master nodes. The number depends on the plan used to create the cluster.
1 or more	Kubernetes worker nodes. The number depends on the plan used to create the cluster, or the number specified during cluster creation.

Note: For production clusters, three master nodes are required, and a minimum of three worker nodes are required. See Requirements for PKS on vSphere with NSX-T for more information.

NSX-T Logical Switches

When a new Kubernetes cluster is created, PKS creates the following NSX-T logical switches:

Object Number	Object Description
1	Logical switch for Kubernetes master and worker nodes.
1	Logical switch for each Kubernetes namespace: `default`, `kube-public`, `kube-system`, `pks-infrastructure`.
1	Logical switch for the NSX-T load balancer associated with the Kubernetes cluster.

NSX-T Tier-1 Logical Routers

When a new Kubernetes cluster is created, PKS creates the following NSX-T Tier-1 logical routers:

Object Number	Object Description
1	Tier-1 router for Kubernetes master and worker nodes. Name: `cluster-router`.
1	Tier-1 router for each Kubernetes namespace: `default`, `kube-public`, `kube-system`, `pks-infrastructure`.
1	Tier-1 router for the NSX-T load balancer associated with the Kubernetes cluster.

NSX-T Load Balancers

For each Kubernetes cluster created, PKS creates a single instance of a small NSX-T load balancer. This load balancer contains the objects listed in the following table:

Object Number	Object Description
1	Virtual Server (VS) to access Kubernetes control plane API on port 8443.
1	Server Pool containing the 3 Kubernetes master nodes.
1	VS for HTTP Ingress Controller.
1	VS for HTTPS Ingress Controller.

The IP address allocated to each VS is derived from the **Floating IP Pool** that was created for use with PKS. The VS for the HTTP Ingress Controller and the VS for the HTTPS Ingress Controller use the same IP address.

NSX-T DDI/IPAM

For each Kubernetes cluster created, PKS extracts and allocates the following NSX-T subnets from the IP blocks created in preparation for installing PKS with NSX-T:

Object Number	Object Description
1	A /24 subnet from the **Nodes IP Block** will be extracted and allocated for the Kubernetes master and worker nodes.
1	A /24 subnet from the **Pods IP Block** will be extracted and allocated for each Kubernetes namespace: `default`, `kube-public`, `kube-system`, `pks-infrastructure`.

NSX-T Tier-0 Logical Routers

For each Kubernetes cluster created, PKS defines the following NSX-T NAT rules on the Tier-0 logical router:

Object Number	Object Description
1	SNAT rule created for each Kubernetes namespace: `default`, `kube-public`, `kube-system`, `pks-infrastructure` using 1 IP from the **Floating IP Pool** as translated IP address.
1	(NAT topology only) SNAT rule created for each Kubernetes cluster using 1 IP from the **Floating IP Pool** as translated IP address. The Kubernetes cluster subnet is derived from the **Nodes IP Block** using a /24 netmask.

NSX-T Distributed Firewall (DFW) Rules

For each Kubernetes cluster created, PKS defines the following NSX-T distributed firewall rules:

Object Amount	Object Description
1	DFW rule for kubernetes-dashboard: `Source=Kubernetes worker node (hosting the Dashboard Pod); Destination=Dashboard Pod IP; Port: TCP/8443; Action: allow`
1	DFW rule for kube-dns: `Source=Kubernetes worker node (hosting the DNS Pod); Destination=DNS Pod IP; Port: TCP/8081 and TCP/10054; Action: allow`

Please send any feedback you have to pks-feedback@pivotal.io.

Planning, Preparing, and Configuring NSX-T for PKS

Page last updated:

Before you install PKS on vSphere with NSX-T integration, you must prepare your NSX-T environment. Complete all of the steps listed in the order presented to manually create the NSX-T environment for PKS.

Step 1: Plan Network Topology, Subnets, and IP Blocks

Plan NSX-T Deployment Topology

Review vSphere with NSX-T Version Requirements and Hardware Requirements for PKS on vSphere with NSX-T.

Review the Deployment Toplogies for PKS on vSphere with NSX-T, and the NSX-T Data Center documentation to ensure that your chosen network topology will enable the following communications:

- vCenter, NSX-T components, and ESXi hosts must be able to communicate with each other.
- The BOSH Director VM must be able to communicate with vCenter and the NSX Manager.
- The BOSH Director VM must be able to communicate with all nodes in all Kubernetes clusters.
- Each PKS-provisioned Kubernetes cluster deploys the NSX-T Node Agent and the Kube Proxy that run as BOSH-managed processes on each worker node.

In addition, the NSX-T Container Plugin (NCP) runs as a BOSH-managed process on the Kubernetes master node. In a multi-master PKS deployment, the NCP process runs on all master nodes. However, the process is active only on one master node. If the NCP process on an active master is unresponsive, BOSH activates another NCP process. Refer to the NCP documentation for more information.

Plan Network CIDRs

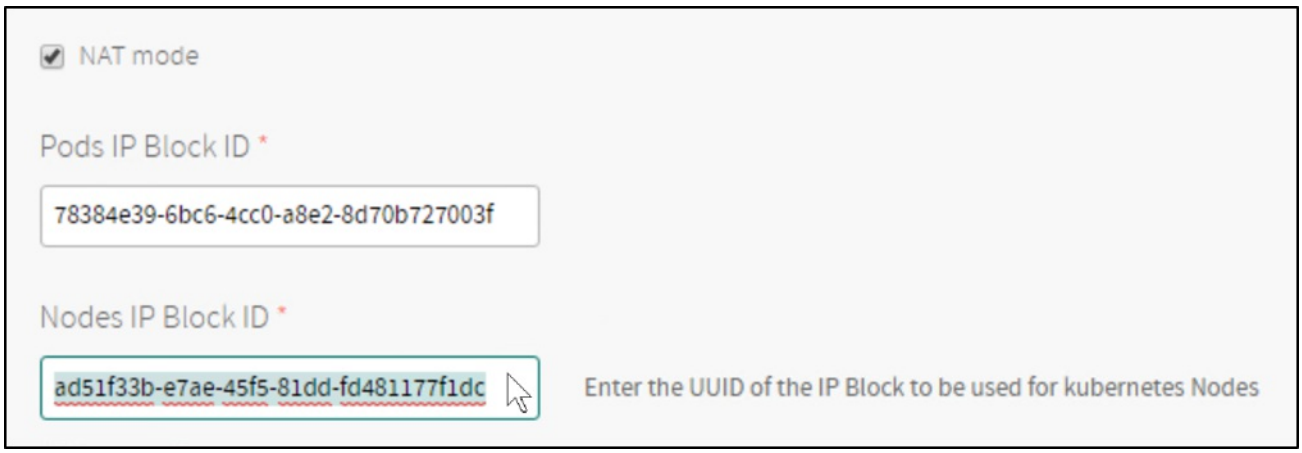
Before you install PKS on vSphere with NSX-T, you should plan for the CIDRs and IP blocks that you are using in your deployment.

Plan for the following network CIDRs in the IPv4 address space according to the instructions in the VMware NSX-T documentation.

- **VTEP CIDRs**: One or more of these networks host your GENEVE Tunnel Endpoints on your NSX Transport Nodes. Size the networks to support all of your expected Host and Edge Transport Nodes. For example, a CIDR of `192.168.1.0/24` provides 254 usable IPs.
- **PKS MANAGEMENT CIDR**: This small network is used to access PKS management components such as Ops Manager, BOSH Director, the PKS Service VM, and the Harbor Registry VM (if deployed). For example, a CIDR of `10.172.1.0/28` provides 14 usable IPs. For the No-NAT deployment topologies, this is a corporate routable subnet /28. For the NAT deployment topology, this is a non-routable subnet /28, and DNAT needs to be configured in NSX-T to access the PKS management components.
- **PKS LB CIDR**: This network provides your load balancing address space for each Kubernetes cluster created by PKS. The network also provides IP addresses for Kubernetes API access and Kubernetes exposed services. For example, `10.172.2.0/24` provides 256 usable IPs. This network is used when creating the `ip-pool-vips` described in Creating NSX-T Objects for PKS, or when the services are deployed. You enter this network in the **Floating IP Pool ID** field in the **Networking** pane of the PKS tile.

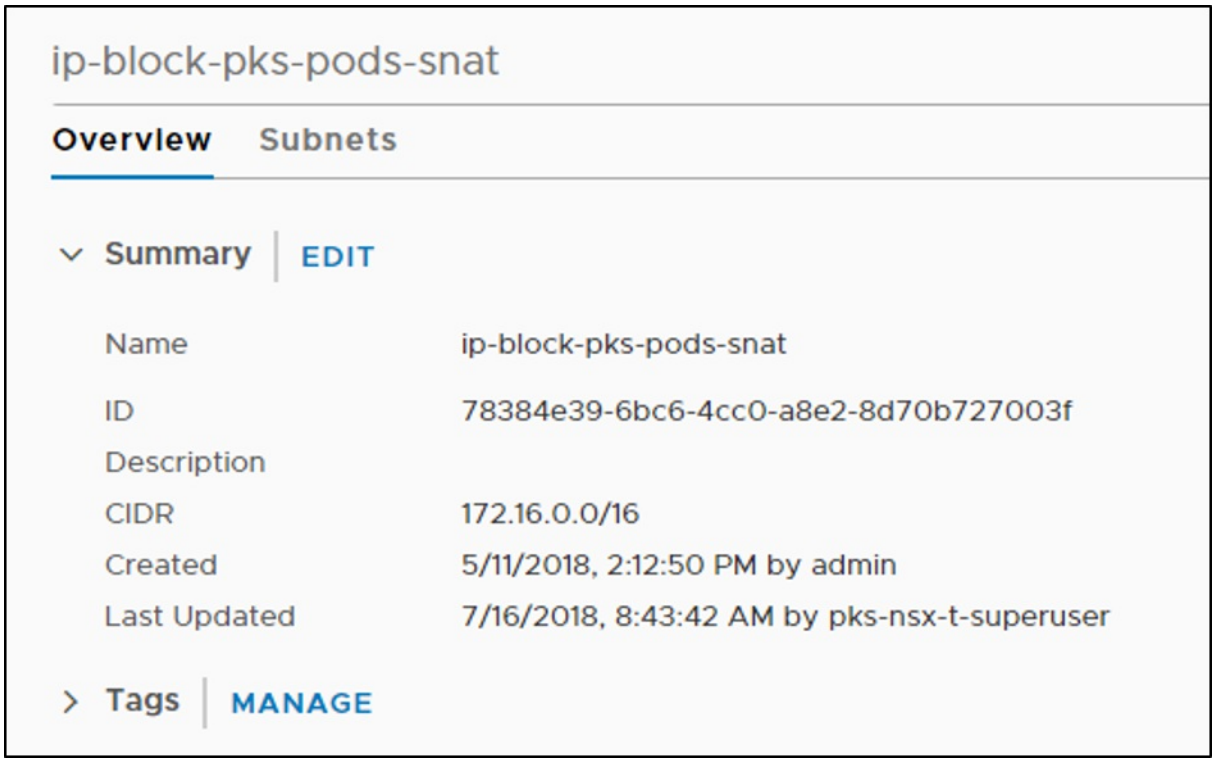
Plan IP Blocks

When you install PKS on NSX-T, you are required to specify the **Pods IP Block ID** and **Nodes IP Block ID** in the **Networking** pane of the PKS tile. These IDs map to the two IP blocks you must configure in NSX-T: the Pods IP Block for Kubernetes pods, and the Node IP Block for Kubernetes nodes (VMs). For more information, see the Networking section of *Installing PKS on vSphere with NSX-T Integration*.



Pods IP Block

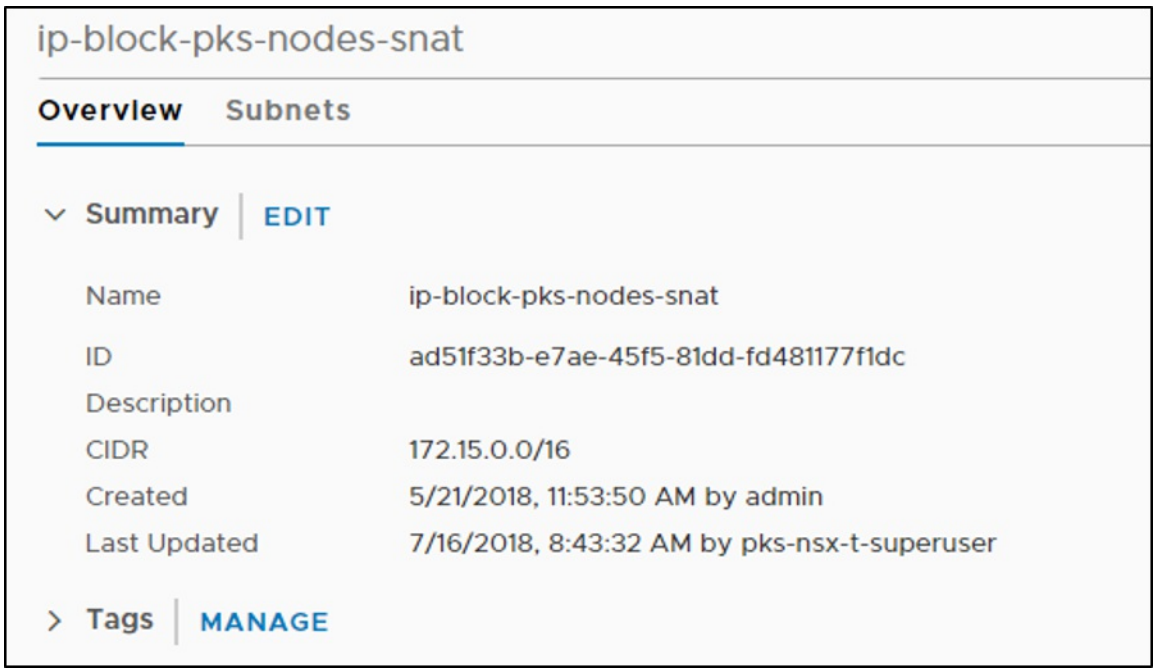
Each time a Kubernetes namespace is created, a subnet from the **Pods IP Block** is allocated. The subnet size carved out from this block is /24, which means a maximum of 256 pods can be created per namespace. When a Kubernetes cluster is deployed by PKS, by default 3 namespaces are created. Often additional namespaces will be created by operators to facilitate cluster use. As a result, when creating the **Pods IP Block**, you must use a CIDR range larger than /24 to ensure that NSX has enough IP addresses to allocate for all pods. The recommended size is /16. For more information, see Creating NSX-T Objects for PKS.



Note: By default, **Pods IP Block** is a block of non-routable, private IP addresses. After you deploy PKS, you can define a network profile that specifies a routable IP block for your pods. The routable IP block overrides the default non-routable **Pods IP Block** when a Kubernetes cluster is deployed using that network profile. For more information, see Routable Pods in *Using Network Profiles (NSX-T Only)*.

Nodes IP Block

Each Kubernetes cluster deployed by PKS owns a /24 subnet. To deploy multiple Kubernetes clusters, set the **Nodes IP Block ID** in the **Networking** pane of the PKS tile to larger than /24. The recommended size is /16. For more information, see Creating NSX-T Objects for PKS.



Note: You can use a smaller nodes block size for no-NAT environments with a limited number of routable subnets. For example, /20 allows up to 16 Kubernetes clusters to be created.

Reserved IP Blocks

The PKS Management Plane must not use the use 172.17.0.0/16 subnet. This restriction applies to all virtual machines (VMs) deployed during the PKS installation process, including the PKS control plane, Ops Manager, BOSH Director, and Harbor Registry.

In addition, do not use any of the IP blocks listed below for Kubernetes master or worker node VMs, or for Kubernetes pods. If you create Kubernetes clusters with any of the blocks listed below, the Kubernetes worker nodes cannot reach Harbor or internal Kubernetes services.

The Docker daemon on the Kubernetes worker node uses the subnet in the following CIDR range. Do not use IP addresses in the following CIDR range:

- 172.17.0.1/16
- 172.18.0.1/16
- 172.19.0.1/16
- 172.20.0.1/16
- 172.21.0.1/16
- 172.22.0.1/16

If PKS is deployed with Harbor, Harbor uses the following CIDR ranges for its internal Docker bridges. Do not use IP addresses in the following CIDR range:

- 172.18.0.0/16
- 172.19.0.0/16
- 172.20.0.0/16
- 172.21.0.0/16
- 172.22.0.0/16

Each Kubernetes cluster uses the following subnet for Kubernetes services. Do not use the following IP block for the Nodes IP Block:

- 10.100.200.0/24

Step 2: Deploy NSX Manager

Deploy the NSX Manager Unified Appliance. For instructions, see Deploy the NSX Manager.

Step 3: Deploy NSX Controllers

Deploy one or more NSX Controllers. You must deploy at least one NSX Controller for PKS; three NSX Controllers are recommended. For instructions, see Deploy NSX Controllers.

Step 4: Create NSX Clusters

Create NSX Clusters for the Management Plane and Control Plane. For instructions, see Create NSX Clusters.

Step 5: Deploy NSX Edge Nodes

Deploy two or more NSX Edge Nodes. Edge Nodes for PKS run load balancers for PKS API traffic, load balancer services for Kubenetes pods, and ingress controllers for Kubernetes pods. For instructions, see Deploy NSX Edge Nodes.

PKS supports active/standby Edge Node failover and requires at least two Edge Nodes. In addition, PKS requires the Edge Node Large VM (8 vCPU, 16 GB of RAM, and 120 GB of storage). The default size of the LB provisioned for PKS is small. You can customize this after deploying PKS using Network Profiles.

The table below lists the maximum number of load balancers per Edge Node form factor.

Edge Node Type	LB Small Max	LB Medium Max	LB Large Max	Supported by PKS
Edge VM Small	0	0	0	No
Edge VM Medium	1	0	0	No
Edge VM Large	40	4	0	Yes
Edge Bare Metal	750	100	7	Yes

Keep in mind the following requirements for NSX Edge Nodes with PKS:

- PKS requires the NSX-T Edge Node large VM (8 vCPU and 16 GB of RAM) or the bare metal Edge Node. For more information, see Hardware requirements for PKS on vSphere with NSX-T.
- The default load balancer deployed by NSX-T for a PKS-provisioned Kubernetes cluster is the small load balancer. The size of the load balancer can be customized using Network Profiles.
- Edge Node VMs can only be deployed on Intel-based ESXi hosts.
- The large load balancer requires a bare metal Edge Node.
- For high-availability Edge Nodes are deployed as pairs within an Edge Cluster. The minimum number of Edge Nodes per Edge Cluster is 2; the maximum is 10. PKS supports active/standby mode only. In standby mode, the standby LB is not available for use while the active LB is active. To determine the maximum number of load balancers per Edge Cluster, multiply the maximum number of LBs for the Edge Node type by the number of Edge Nodes and divide by 2. For example, with 10 Edge VM Large nodes in an Edge Cluster, you can have up to 200 small LB instances (40 x 10 / 2), or up to 20 medium LB instances (4 x 10 / 2).
- PKS deploys a virtual server for each load balancer instance. For service of type load balancer, it is one virtual server per service. There are two global virtual servers deployed for ingress resources (HTTP and HTTPS). And there is one global virtual server for the PKS API. For more information, see Defining Network Profiles.

Step 6: Register NSX Edge Nodes

Register NSX Edge Nodes with the NSX Manager. For instructions, see Register NSX Edge Nodes.

Step 7: Enable VIB Repository Service

The VIB repository service provides access to native libraries for NSX Transport Nodes. VIB must be enabled before you proceed further with deploying NSX. For instructions, see Enable VIB Repository Service on NSX Manager.

Step 8: Create TEP IP Pool

Create Tunnel Endpoint IP Pool (TEP IP Pool) within the usable range of the **VTEP CIDR** that was defined in [preparation for installing NSX-T](#plan-cidrs). The TEP IP Pool is used for NSX Transport Nodes. For instructions, see Create TEP IP Pool.

Step 9: Create Overlay Transport Zone

Create an NSX Overlay Transport Zone (TZ-Overlay) for PKS Control Plane services and Kubernetes Cluster deployment overlay networks. For instructions, see Create Overlay TZ.

Step 10: Create VLAN Transport Zone

Create an NSX VLAN Transport Zone (TZ-VLAN) for NSX Edge uplinks (ingress/egress) for PKS-managed Kubernetes clusters. For instructions, see Create VLAN TZ.

Step 11: Create Uplink Profile for Edge Nodes

Create an NSX Uplink Profile for NSX Edge Nodes to be used with PKS. For instructions, see Create Uplink Profile for Edge Nodes.

Step 12: Create Transport Edge Nodes

Create NSX Edge Transport Nodes, which allow Edge Nodes to exchange traffic for virtual networks among other NSX nodes. For instructions, see Create Transport Edge Nodes.

Step 13: Create Edge Cluster

Create an NSX Edge Cluster and add each NSX Edge Transport Node to the Edge Cluster. For instructions, see Create Edge Cluster.

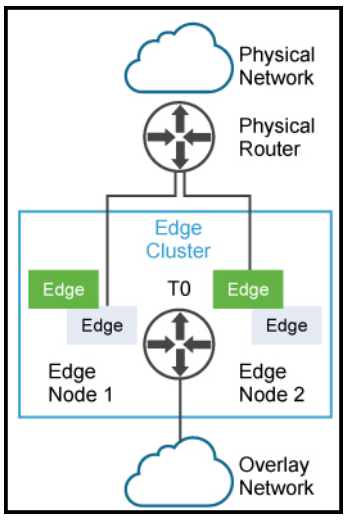
Step 14: Create T0 Logical Router for PKS

NSX Tier-0 Logical Routers are used to route data between the NSX-T virtual network and the physical network. For instructions, see Create T0 Router.

Step 15: Configure NSX Edge for High Availability (HA)

Configure NSX Edge for high availability (HA) using Active/Standby mode to support failover, as shown in the following figure. For instructions, see Configure Edge HA.

Note: If the T0 Router is not configured for HA as decribed in Configure Edge Nodes for HA, failover to the standby Edge Node will not occur.



Step 16: Prepare ESXi Hosts for PKS Compute Plane

An NSX Transport Node allows NSX Nodes to exchange traffic for virtual networks. ESXi hosts dedicated to the PKS Compute Cluster must be prepared as tranport nodes. For instructions, see Prepare Compute Cluster ESXi Hosts.

Note: The Transport Nodes must be placed on free host NICs not already used by other vSwitches on the ESXi host. Use the `VTEPS` IP pool that allows ESXi hosts to route and communicate with each other, as well as other Edge Transport Nodes.

Step 17: Create NSX-T Objects for PKS Management Plane

Prepare the vSphere and NSX-T infrastructure for the PKS Management Plane where the PKS, Ops Manager, BOSH Director, and Harbor Registry VMs are deployed. This includes a vSphere resource pool for PKS management components, an NSX Tier-1 (T1) Logical Switch, and an NSX Tier-1 Logical Router and Port. For instructions, see Prepare PKS Management Plane.

If you are using the NAT Topology, create the following NAT rules on the T0 Router. For instructions, see Prepare Management Plane.

Type	For
DNAT	External > Ops Manager
DNAT	External > Harbor (optional)
SNAT	PKS Management Plane > vCenter and NSX-T Manager
SNAT	PKS Management Plane > DNS
SNAT	PKS Management Plane > NTP
SNAT	PKS Management Plane > LDAP/AD (optional)
SNAT	PKS Management Plane > ESXi

Step 18: Create NSX-T Objects for PKS Compute Plane

Create Resource Pools for AZ-1 and AZ-2, which map to the Availability Zones you will create when you configure BOSH Director and reference when you install the PKS tile. In addition, create SNAT rules on the T0 router:

- One for K8s Master Nodes (hosting NCP) to reach the NSX-T Manager
- One for Kubernetes Master Node Access to LDAP/AD (optional)

For instructions, see Prepare Compute Plane.

Step 19: Deploy Ops Manager in the NSX-T Environment

Deploy Ops Manager 2.3.2+ on the NSX-T Management Plane network. For instructions, see Deploy Ops Manager on vSphere with NSX-T.

Step 20: Generate NSX Manager Certificate

Generate the CA Cert for the NSX Manager and import the certificate to NSX Manager. For instructions, see Generate the NSX Manager CA Cert.

Step 21: Configure BOSH Director for vSphere with NSX-T

Create BOSH availability zones (AZs) that map to the Management and Compute resource pools in vSphere, and the Management and Control plane networks in NSX-T. For instructions, see Configure BOSH Director for vSphere with NSX-T.

Step 22: Generate NSX Manager Principal Identity Certificate

Generate the NSX Manager Super User Principal Identity Certificate and register it with the NSX Manager using the NSX API. For instructions, see Generate the NSX Manager PI Cert.

Step 23: Create NSX-T Objects for PKS

Create IP blocks for the node networks and the pod networks. The subnets for both nodes and pods should have a size of 256 (/16). See Plan IP Blocks and Reserved IP Blocks for details.

In addition, create a Floating IP Pool from which to assign routable IP addresses to components. This network provides your load balancing address space for each Kubernetes cluster created by PKS. The network also provides IP addresses for Kubernetes API access and Kubernetes exposed services.

These network objects are required to configure the PKS tile for NSX-T networking. For instructions, see Create NSXT Object for PKS.

Step 24: Install PKS on vSphere with NSX-T

At this point your NSX-T environment is prepared for PKS installation using the PKS tile in Ops Manager. For instructions, see Installing PKS on vSphere with NSX-T.

Step 25: Install Harbor Harbor Registry for PKS

The VMware Harbor Registry is recommended for PKS. Install Harbor in the NSX Management Plane with other PKS components (PKS API, Ops Manager, and BOSH). For instructions, see Installing Harbor Registry on vSphere with NSX-T in the PKS Harbor documentation.

If you are using the NAT deployment topology for PKS, create a DNAT rule that maps the private Harbor IP address to a routable IP address from the floating IP pool on the PKS management network. See Create DNAT Rule.

Step 26: Perform Post-Installation NSX-T Configurations as Necessary

Once PKS is installed, you may want to perform additional NSX-T configurations to support customization of Kubernetes clusters at deployment time, such as:

- Configuring an HTTTP Proxy to proxy outgoing HTTP/S traffic from NCP, PKS, BOSH, and Ops Manager to vSphere infrastructure components (vCenter, NSX Manager)
- Defining Network Profiles to customize NSX-T networking objects, such as load balancer size, custom Pods IP Block, routable Pods IP Block, configurable CIDR range for the Pods IP Block, custom Floating IP block, and more.
- Configuring Multiple Tier-0 Routers to support customer/tenant isolation

Deploying NSX-T for PKS

Page last updated:

To deploy NSX-T for PKS, complete the following set of procedures, in the order presented.

Note: The instructions provided in this topic are based on NSX-T v2.3.

Before you begin this procedure, ensure that you have successfully completed all preceding steps for installing PKS on vSphere with NSX-T, including:

- vSphere with NSX-T Version Requirements
- Hardware Requirements for PKS on vSphere with NSX-T
- NSX-T Deployment Topologies for PKS
- Preparing to Deploy PKS with NSX-T on vSphere

Step 1: Deploy NSX Manager

The NSX Manager is provided as an OVA file named **NSX Unified Appliance** that you import into your vSphere environment and configure.

Complete either of the following procedures to deploy the NSX Manager appliance:

- Deploy NSX Manager using the vSphere client
- Deploy NSX Manager using the ovftool CLI

To verify deployment of the NSX Manager:

1. Power on the NSX Manager VM.
2. Ping the NSX Manager VM. Get the IP address for the NSX Manager from the **Summary** tab in vCenter. Verify that you can ping the host. For example, run `ping 10.196.188.21`.
3. SSH to the VM. Use the IP address for the NSX Manager to remotely connect using SSH. From Unix hosts use the command `ssh admin@IP_ADDRESS_OF_NSX_MANAGER`. For example, run `ssh admin@10.196.188.21`. On Windows use Putty and provide the IP address. Enter the CLI user name and password that you defined during OVA import.
4. Review NSX CLI usage. Once you are logged into the NSX Manager VM, enter `?` to view the command usage and options for the NSX CLI.
5. Connect to the NSX Manager web interface using a supported browser at the URL `https://IP_ADDRESS_OF_NSX_MANAGER`. For example, `https://10.16.176.10`.

Step 2: Deploy NSX Controllers

The NSX Controller provides communications for NSX-T components.

You must deploy at least one NSX Controller for PKS. Three NSX Controllers are recommended.

Complete either of the following procedures to deploy an NSX Controller:

- Deploy NSX Controllers using the vSphere client
- Deploy NSX Controllers using the ovftool CLI

To verify deployment of the NSX Controller:

1. Power on the NSX Controller VM.
2. Ping the NSX Controller VM. Get the IP address for the NSX Controller from the **Summary** tab in vCenter. Make sure you use a routable IP. If necessary click **View all X IP addresses** to reveal the proper IP address. Verify that you can ping the Controller host. For example, run `ping 10.196.188.22`.
3. SSH to the VM. Use the IP address for the NSX Controller to remotely connect using SSH. From Unix hosts use the command `ssh admin@IP_ADDRESS_OF_NSX_CONTROLLER`. For example, run `ssh admin@10.196.188.22`. On Windows use Putty and provide the IP address. Enter the CLI admin user name and password that you defined during installation.

4. Review NSX CLI usage. After you are logged into the NSX Controller VM, enter `?` to view the command usage and options for the NSX CLI.

Note: Repeat the deployment and verification procedure for each NSX Controller you intend to use for PKS.

Step 3: Create NSX Clusters (Management and Control)

In this section you create NSX Clusters for the PKS Management Plane and Control Plane.

1. Complete this procedure to create the NSX Management Cluster: Join NSX Controllers with the NSX Manager.
2. Complete this procedure to create the NSX Control Cluster: Initialize Control Cluster.
3. If you are deploying more than one NSX Controller, complete this procedure: Join Additional NSX Controllers with the Cluster Master.

To verify the creation of NSX Clusters:

1. Verify that the NSX Controller is `Connected` to the NSX Manager:

```
NSX-CONTROLLER-1> get managers
```

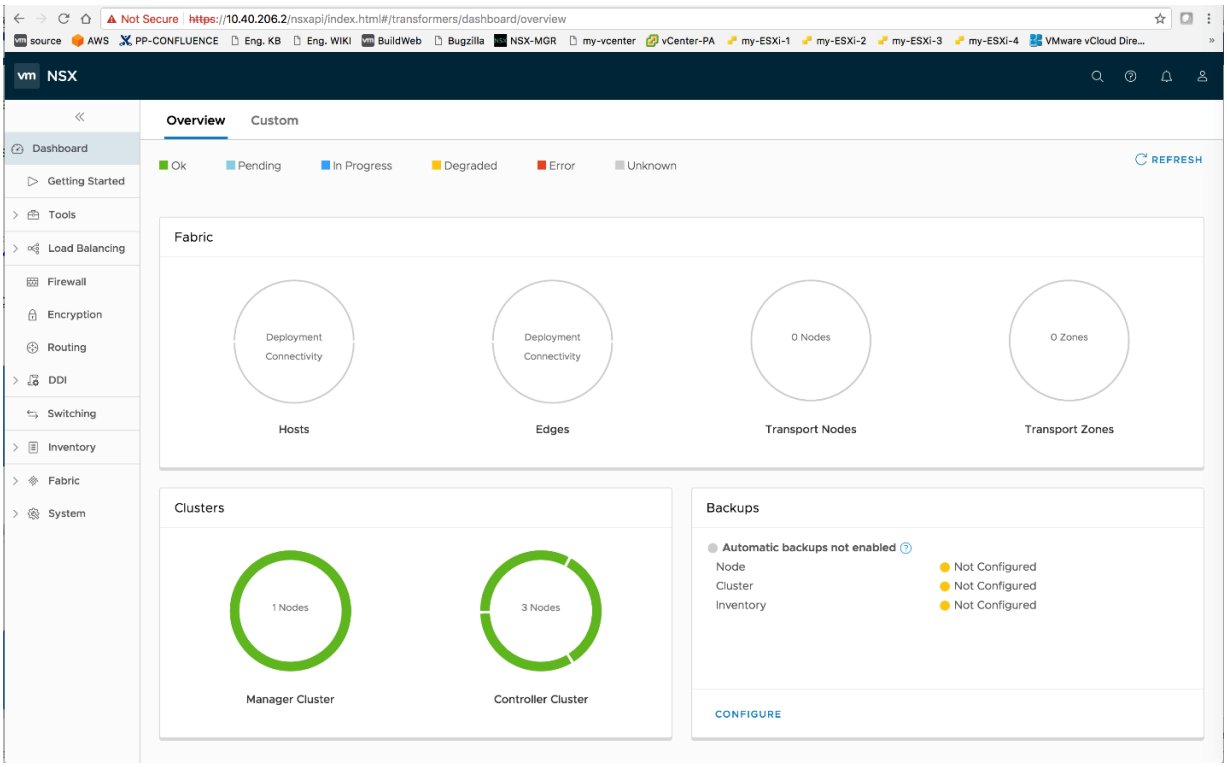
2. Verify that the status of the Control Cluster is `active`:

```
NSX-CONTROLLER-1> get control-cluster status
```

3. Verify that the Management Cluster is `STABLE`:

```
NSX-MGR-1-1-0> get management-cluster status
```

4. Verify the configuration of the NSX Clusters.
   - Connect to the NSX Manager web interface using a supported browser at the URL `https://IP_ADDRESS_OF_NSX_MANAGER`. For example, `https://10.16.176.10`.
   - Log in using your admin credentials.
   - Select **Dashboard > System > Overview**.
   - Confirm that the status of the NSX Manager and each NSX Controller is green.



Step 4: Deploy NSX Edge Nodes

Edge Nodes provide the bridge between the virtual network environment implemented using NSX-T and the physical network. Edge Nodes for PKS run load balancers for PKS API traffic, Kubernetes pod LB services, and pod ingress controllers.

PKS supports active/standby Edge Node failover and requires at least two Edge Nodes. In addition, PKS requires the Edge Node Large VM (8 vCPU, 16 GB of RAM, and 120 GB of storage) or the bare metal Edge Node. See Edge Node Requirements in the VMware documentation for details.

Warning: When deploying an Edge Node VM form factor, you must select the large size VM. This enables PKS to deploy Kubernetes clusters correctly. For more information, see [Deploy NSX Edge Nodes](./nsxt-prepare-env.html#nsx-edges).

For information about load balancers, see Scaling Load Balancer Resources in the VMware documentation.

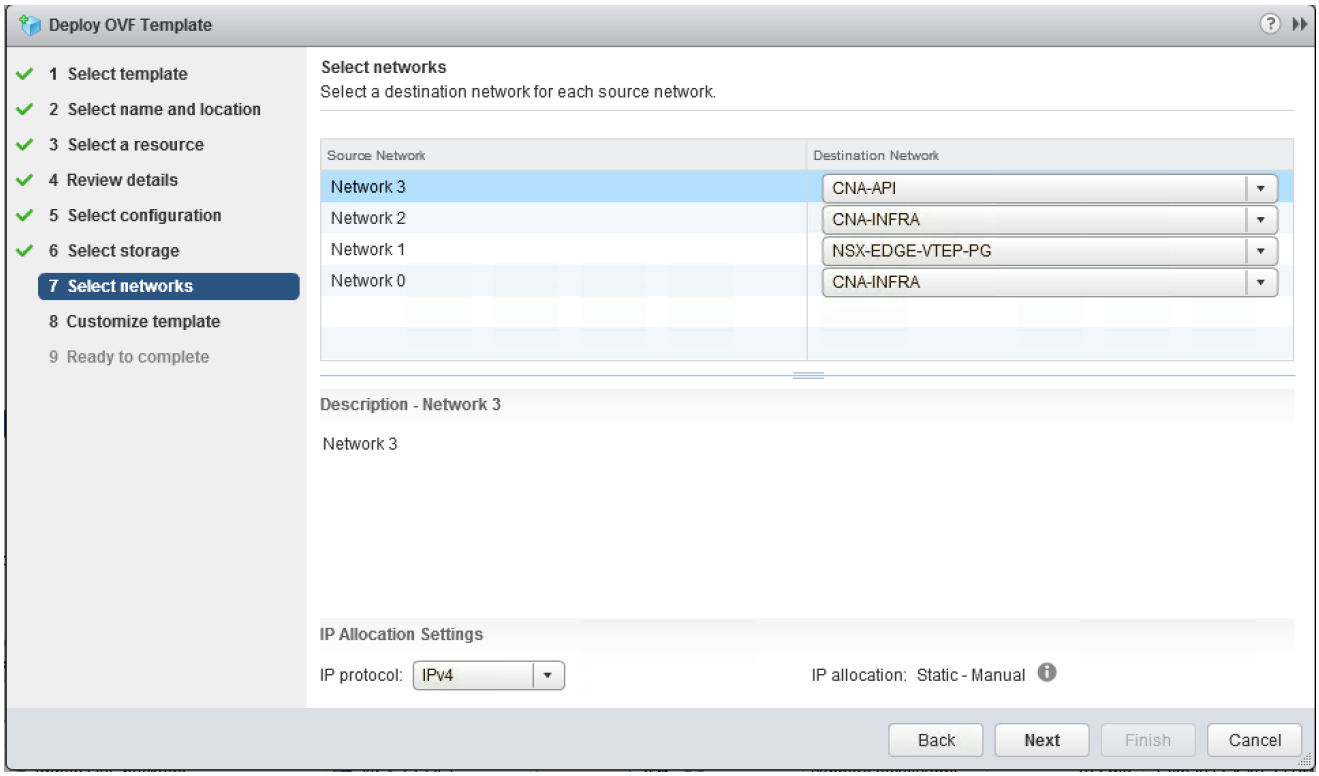
Complete either of the following procedures to deploy an NSX Edge Node:

- Edge Node Installation using vSphere Client
- Edge Node Installation using ofvtool CLI

When deploying the Edge Node, be sure to connect the vNICs of the NSX Edge VMs to an appropriate PortGroup for your environment:

- **Network 0**: For management purposes. Connect the first Edge interface to your environment's PortGroup/VLAN where your Edge Management IP can route and communicate with the NSX Manager.
- **Network 1**: For TEP (Tunnel End Point). Connect the second Edge interface to your environment's PortGroup/VLAN where your GENEVE VTEPs can route and communicate with each other. Your **VTEP CIDR** should be routable to this PortGroup.
- **Network 2**: For uplink connectivity to external physical router. Connect the third Edge interface to your environment's PortGroup/VLAN where your T0 uplink interface is located.
- **Network 3**: Unused (select any port group)

For example:



To verify Edge Node deployment:

1. Power on the Edge Node VM.
2. Ping the Edge VM. Get the IP address for the NSX Manager from the **Summary** tab in vCenter. Verify that you can ping the host by running `ping IP_ADDRESS_OF_NSX_EDGE_NODE`. For example, run `ping 10.196.188.21`.
3. SSH to the Edge VM. Use the IP address for the NSX Manager to remotely connect using SSH. From Unix hosts use the commanc `ssh`

`admin@IP_ADDRESS_OF_NSX_EDGE_NODE`. For example, run `ssh admin@10.196.188.21`. On Windows use Putty and provide the IP address. Enter the CLI admin user name and password that you defined in the **Customize template > Application** section.

4. Review NSX CLI usage. After you are logged into the NSX Manager VM, enter `?` to view the command usage and options for the NSX CLI.

Note: Repeat the deployment and verification process for each NSX Edge Node you intend to use for PKS.

Step 5: Register NSX Edge Nodes with NSX Manager

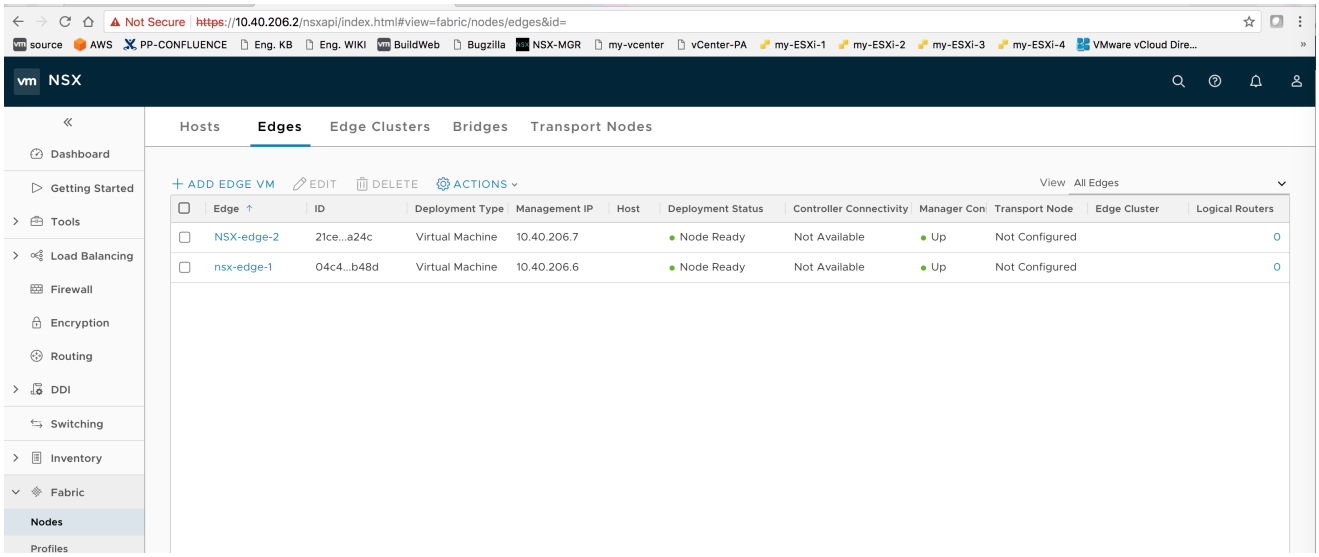
To register an Edge Node with NSX Manager, complete this procedure: Join NSX Edge with the Management Plane.

To verify Edge Node registration with NSX Manager:

1. SSH to the Edge Node and run the following command. Verify that the Status is `Connected`:

```
nsx-edge-1> get managers
```

2. In the NSX Manager Web UI, go to **Fabric > Nodes > Edges**. You should see each registered Edge Node.



Note: Repeat this procedure for each NSX Edge Node you are deploying for PKS.

Step 6: Enable Repository Service on NSX Manager

To enable VIB installation from the NSX Manager repository, the repository service needs to be be enabled in NSX Manager.

1. SSH into NSX Manager by using the command `ssh admin@IP_ADDRESS_OF_NSX_MANAGER` (Unix) or Putty (Windows).

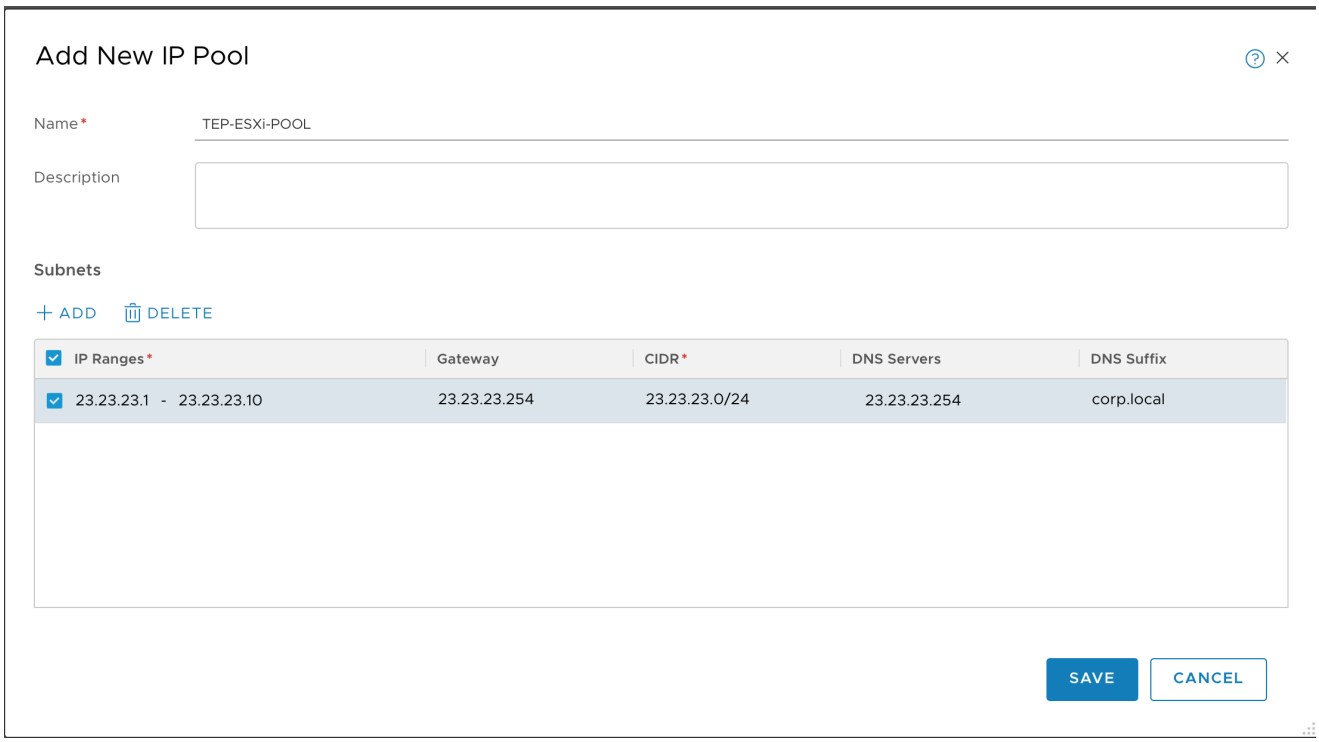
2. Run the following command:

```
nsx-manager> set service install-upgrade enable
```

Step 7: Create TEP IP Pool

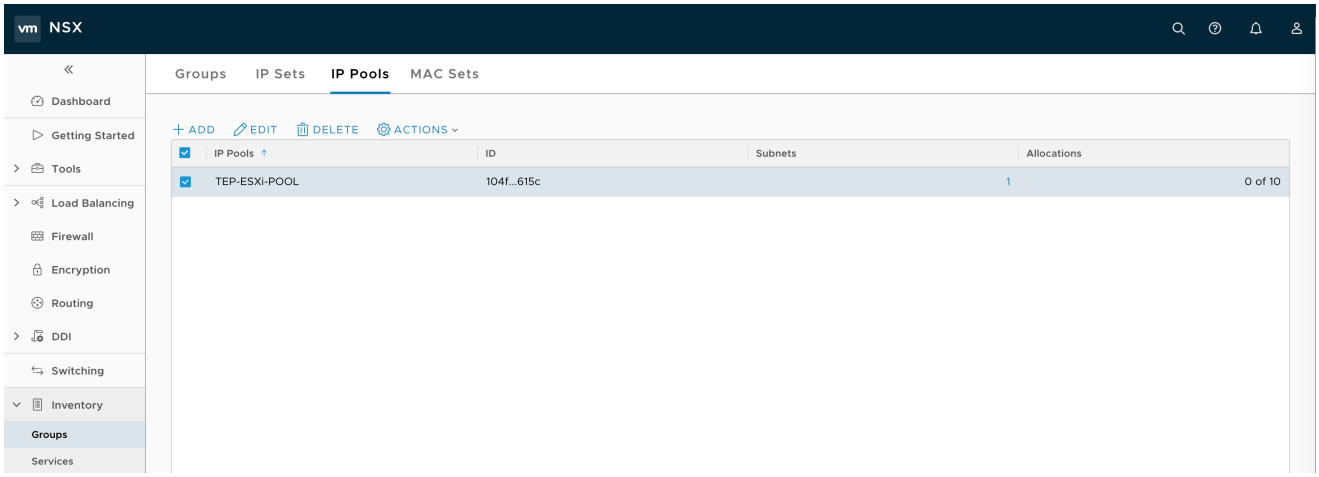
To create the TEP IP Pool, complete this procedure: Create an IP Pool for Tunnel Endpoint IP Addresses.

When creating the TEP IP Pool, refer to the following example:



To verify TEP IP Pool configuration:

1. In NSX Manager, select **Inventory > Groups > IP Pools**.
2. Verify that the TEP IP Pool you created is present.

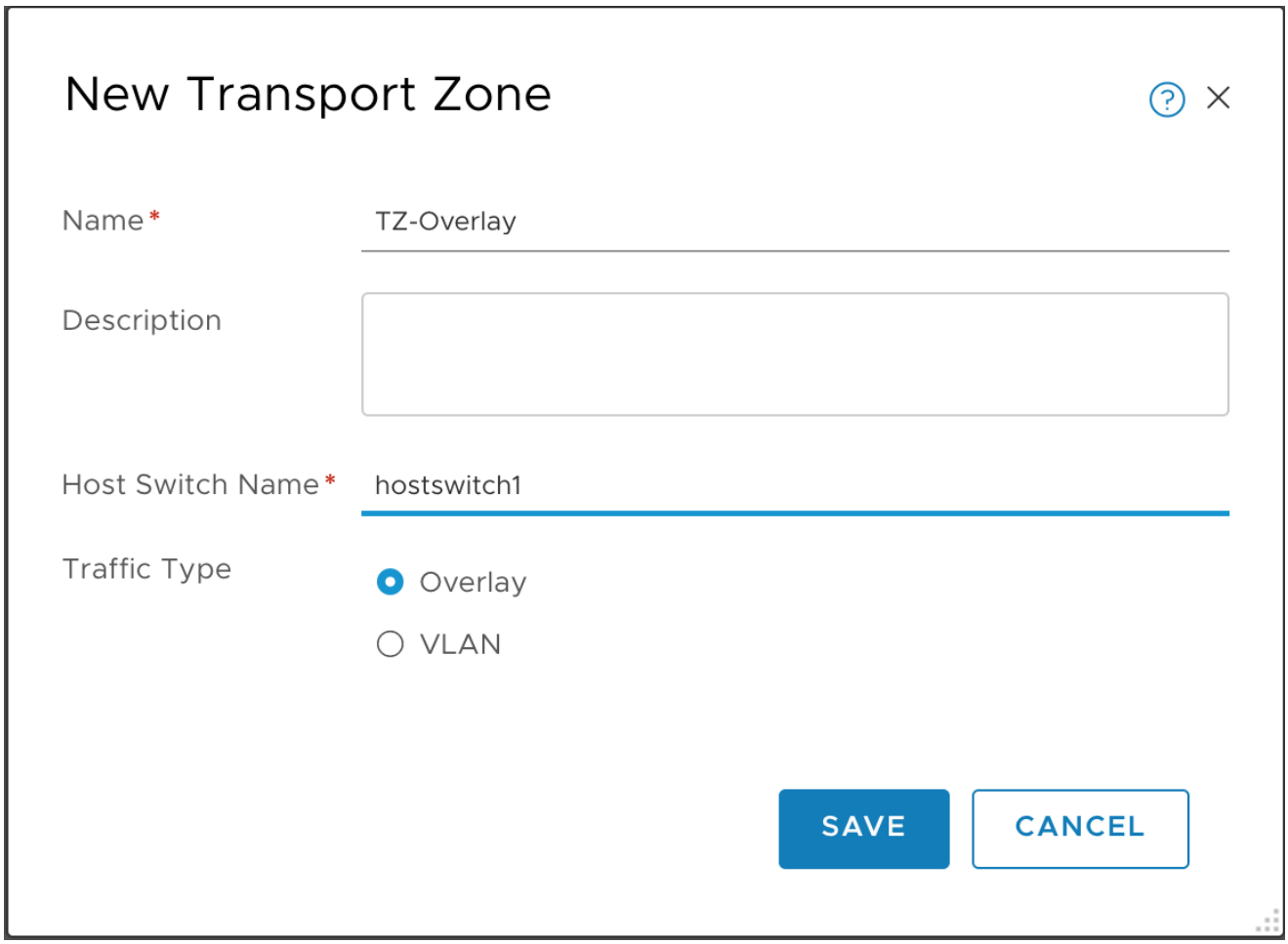


Step 8: Create Overlay Transport Zone

Create an Overlay Transport Zone (`TZ-Overlay`) for PKS control plane services and Kubernetes clusters associated with associated with VDS `hostswitch1` .

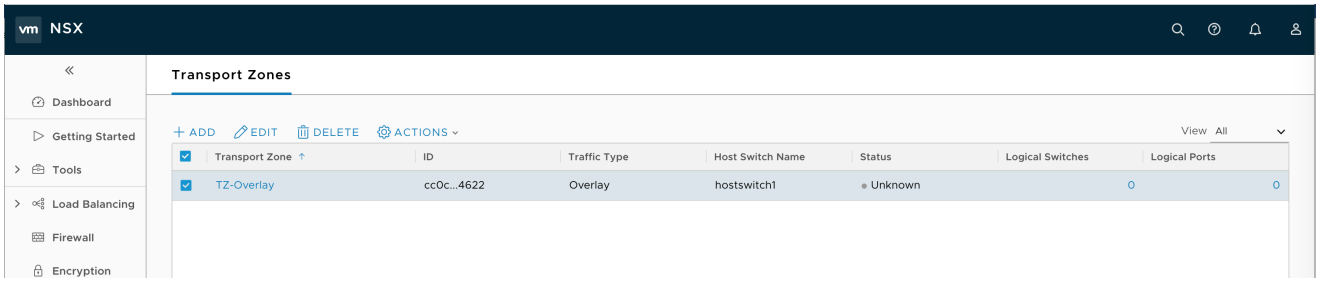
To create TZ-Overlay, complete this procedure: Create Transport Zones.

When creating the TZ-Overlay for PKS, refer to the following example:



To verify TZ-Overlay creation:

1. In NSX Manager select **Fabric > Transport Zones**.
2. Verify that you see the TZ-Overlay transport zone you created:

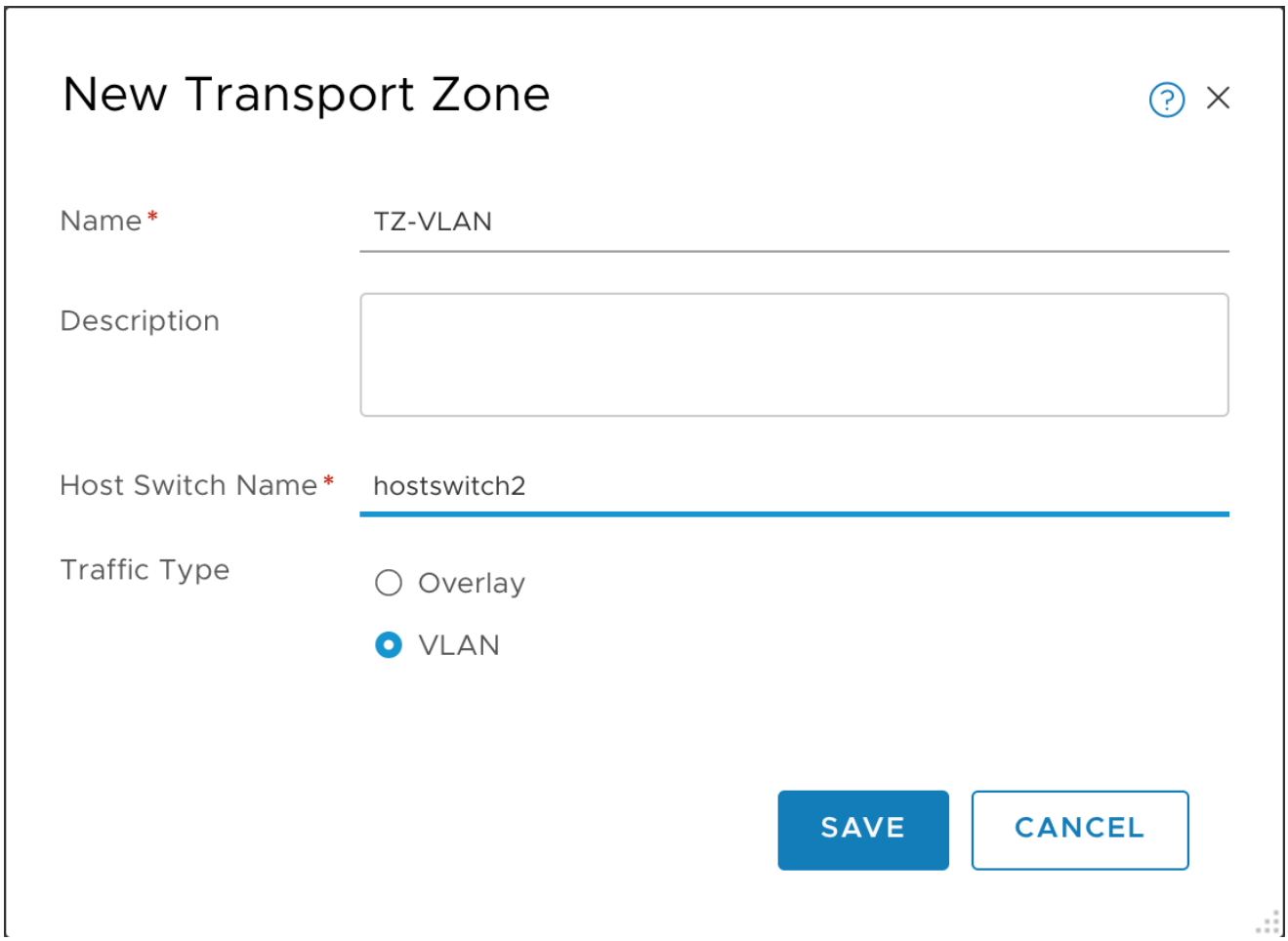


Step 9: Create VLAN Transport Zone

Create the VLAN Transport Zone (`TZ-VLAN`) for NSX Edge Node uplinks (ingress/egress) for PKS Kubernetes clusters associated with VDS `hostswitch2` .

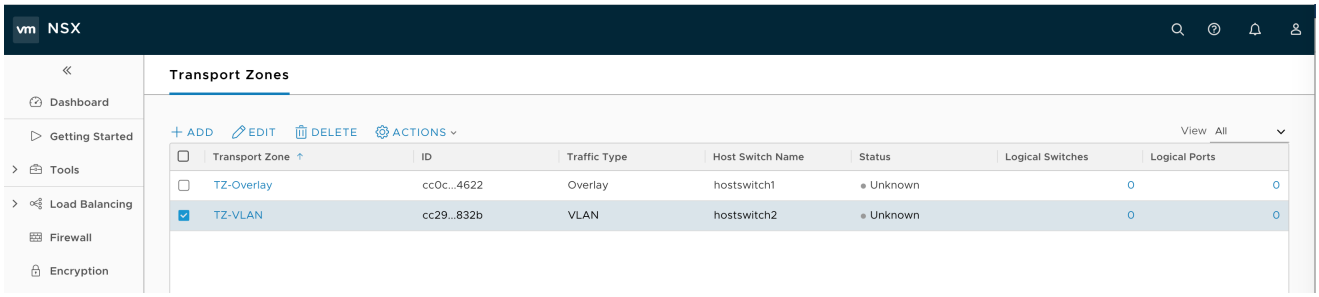
To create TZ-VLAN, complete this procedure: Create Transport Zones.

When creating the TZ-VLAN for PKS, refer to the following example:



To verify TZ-VLAN creation:

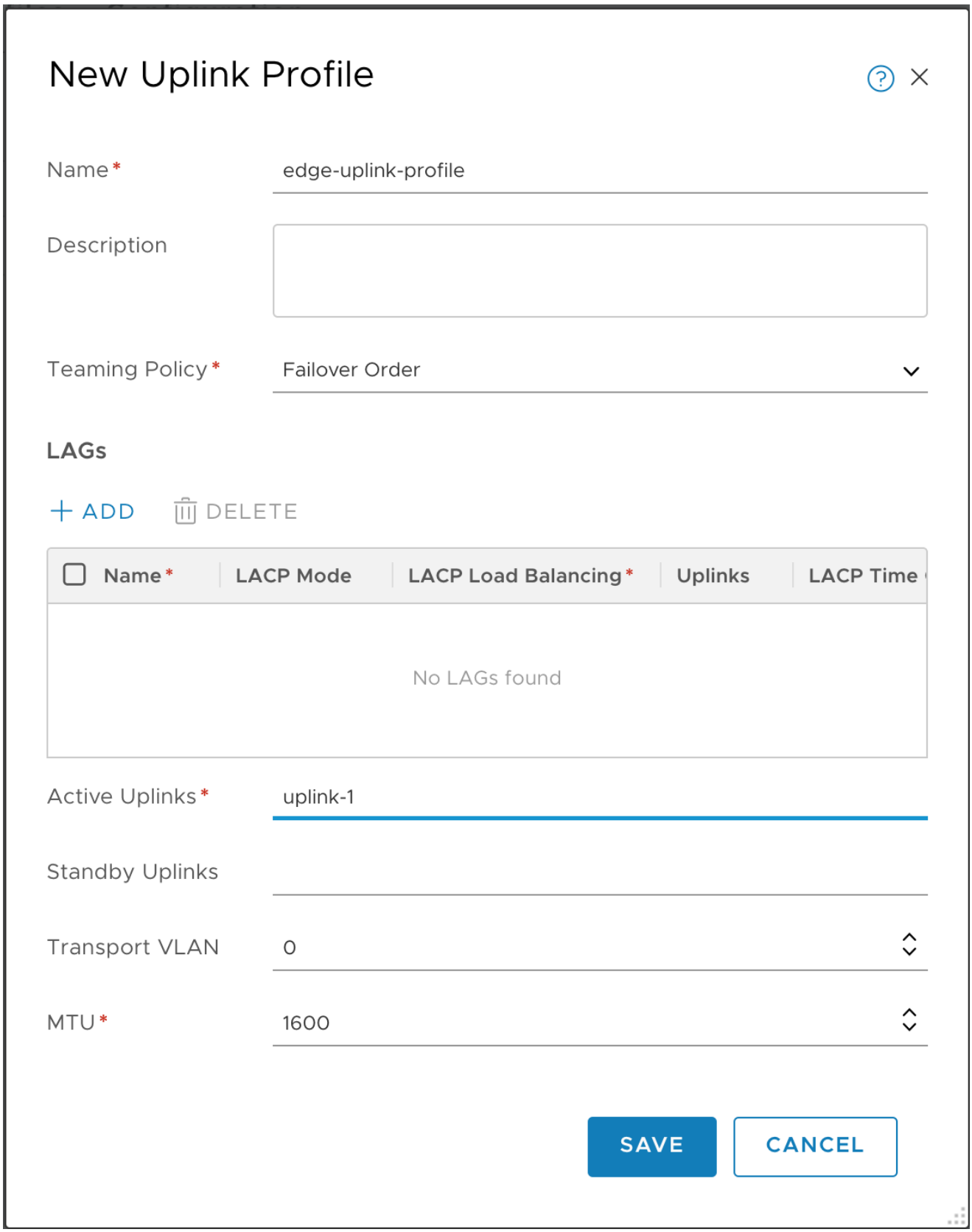
1. In NSX Manager select **Fabric > Transport Zones**.
2. Verify that you see the TZ-VLAN transport zone:



Step 10: Create Uplink Profile for Edge Nodes

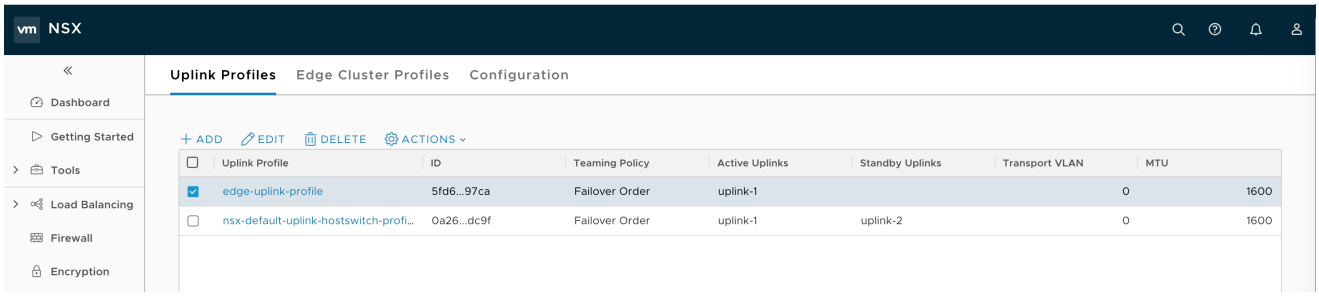
To create an Uplink Profile, complete this procedure: Create an Uplink Profile.

When creating the Uplink Profile for PKS, refer to the following example:



To verify Uplink Profile creation:

1. In NSX Manager select **Fabric > Profiles > Uplink Profiles**.
2. Verify that you see the Edge Node uplink profile you created:



Step 11: Create Edge Transport Nodes

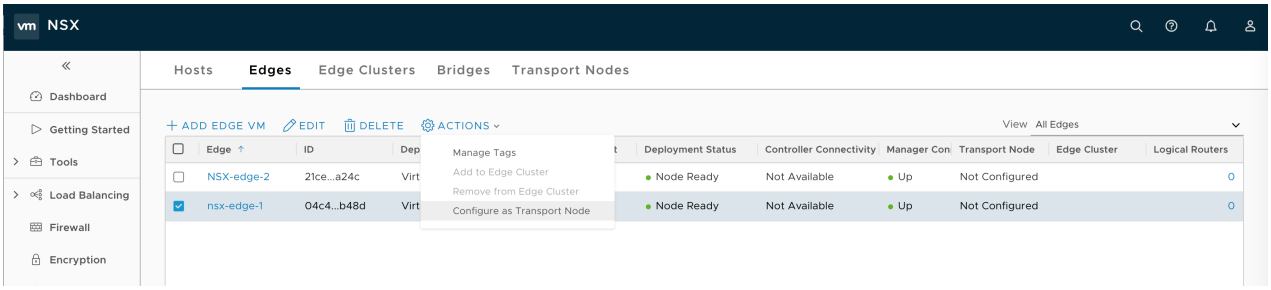
Create NSX Edge Transport Nodes which allow Edge Nodes to exchange virtual network traffic with other NSX nodes.

Be sure to add both the VLAN and OVERLAY NSX Transport Zones to the NSX Edge Transport Nodes and confirm NSX Controller and Manager connectivity. Use the MAC addresses of the Edge VM interfaces to deploy the virtual NSX Edges:

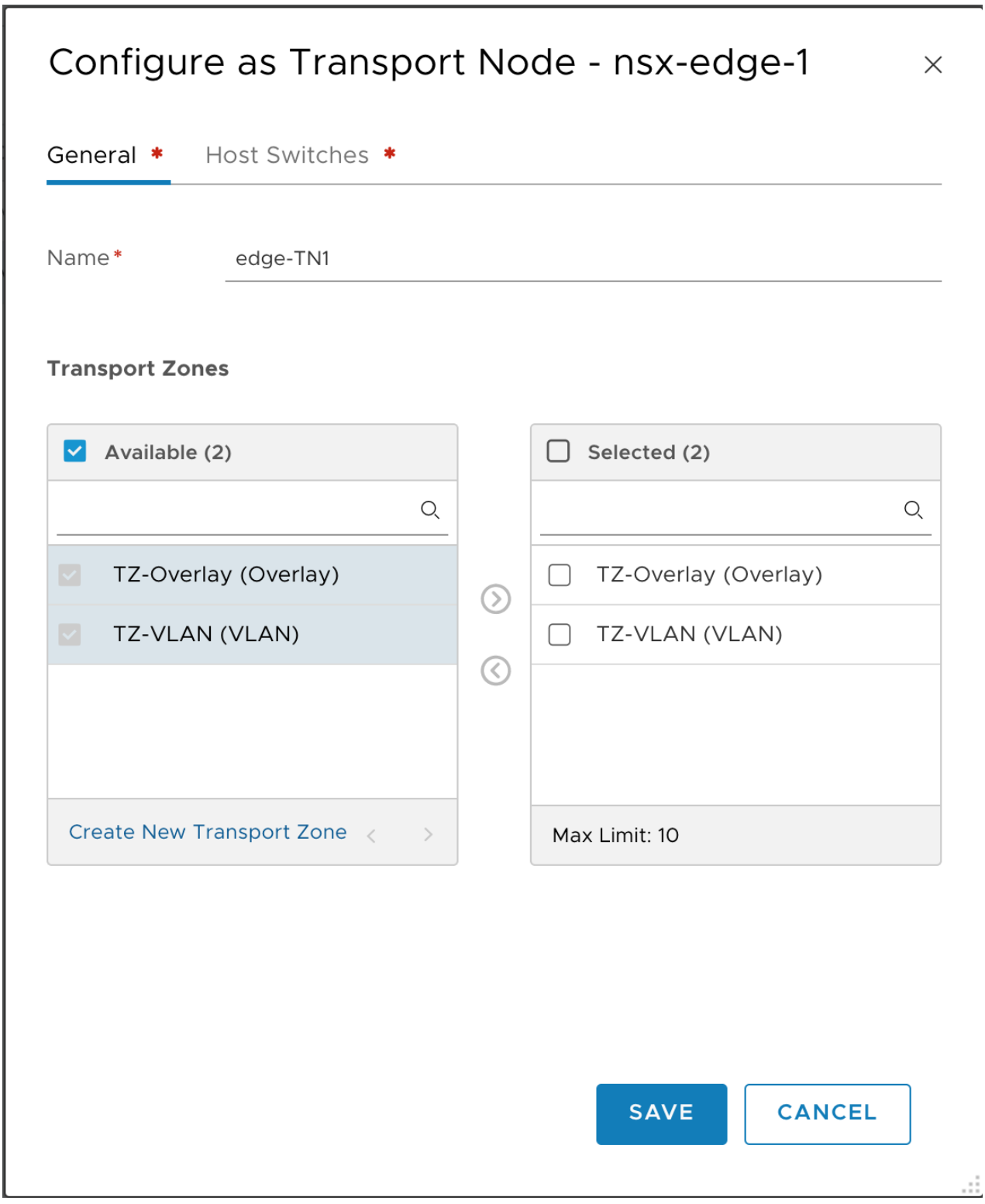
- Connect the OVERLAY N-VDS to the vNIC (`fp-eth#`) that matches the MAC address of the second NIC from your deployed Edge VM.
- Connect the VLAN N-VDS to the vNIC (`fp-eth#`) that matches the MAC address of the third NIC from your deployed Edge VM.

To create an Edge Transport Node for PKS:

1. Log in to NSX Manager (`https://IP_ADDRESS_OF_NSX_MANAGERs`).
2. Go to **Fabric > Nodes > Edges**.
3. Select an Edge Node.
4. Click Actions > **Configure as Transport Node**.



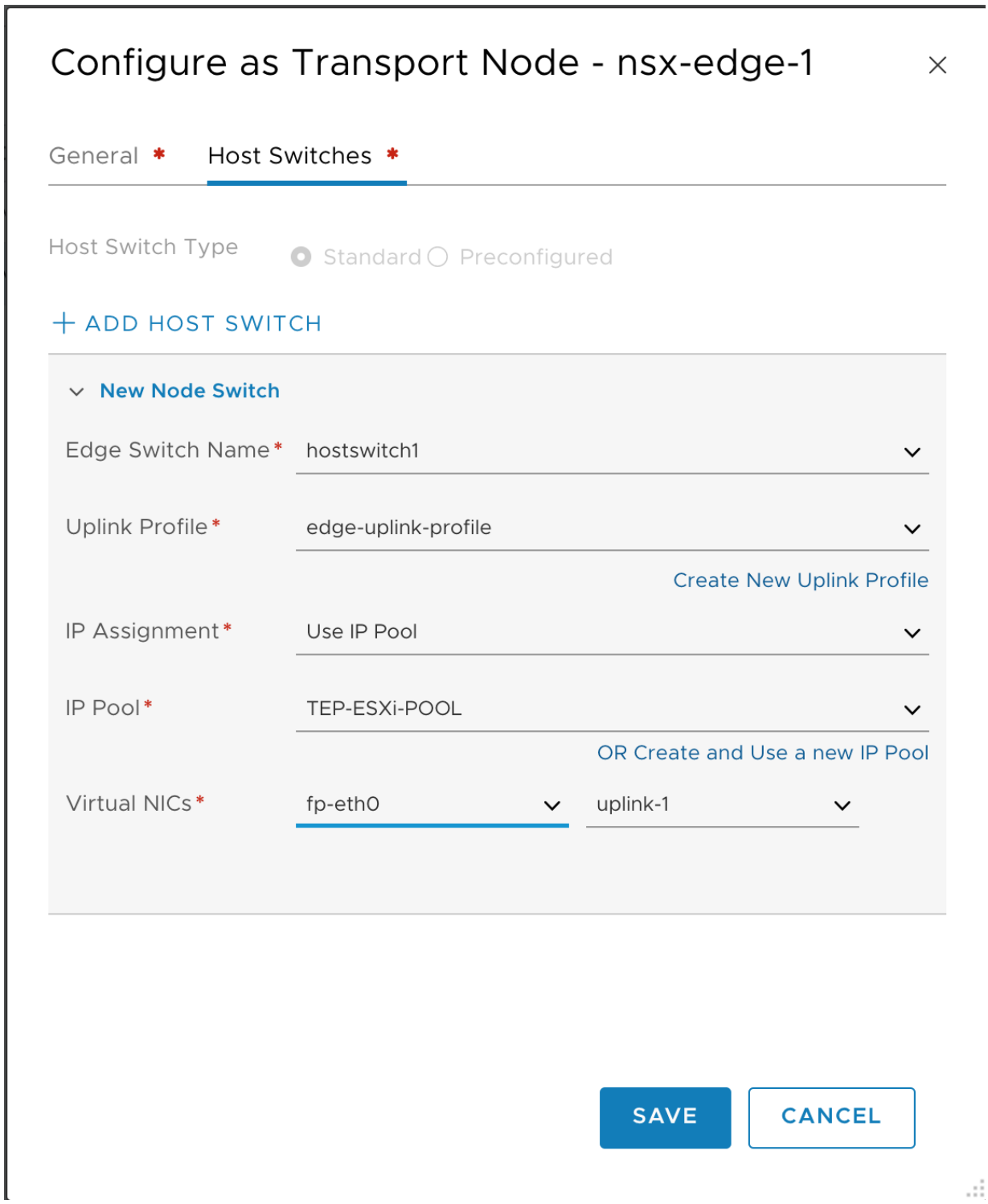
5. In the General tab, enter a name and select both Transport Zones: TZ-Overlay (Overlay) and TZ-VLAN (VLAN).



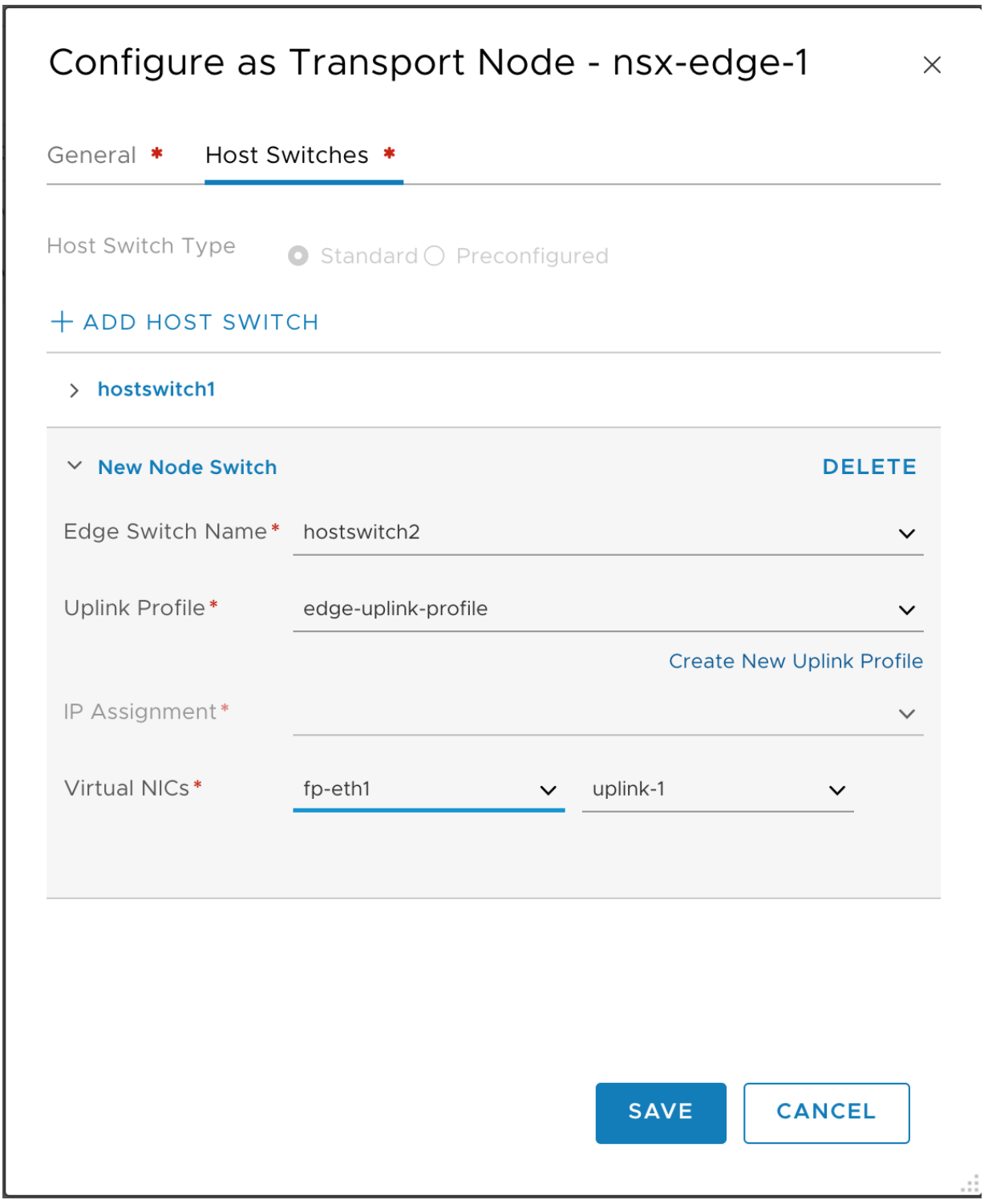
6. Select the **Host Switches** tab.

7. Configure the first transport node switch. For example:

    - **Edge Switch Name:** `hostswitch1`
    - **Uplink Profile:** `edge-uplink-profile`
    - **IP Assignment:** `Use IP Pool`
    - **IP Pool:** `TEP-ESXi-POOL`
    - **Virtual NICs:** `fp-eth0` (corresponds to Edge VM vnic1 (second vnic))



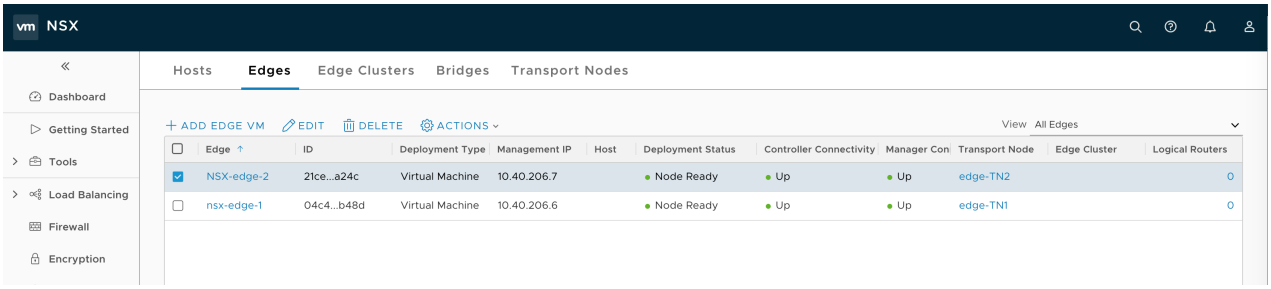
8. Click **Add Host Switch**.
9. Configure the second transport node switch. For example:
    - **Edge Switch Name**: `hostswitch2`
    - **Uplink Profile**: `edge-uplink-profile`
    - **Virtual NICs**: `fp-eth1` (corresponds to Edge VM vnic2 (third vnic))



Note: Repeat this procedure for the second Edge Transport Node (Edge-TN2), as well as additional Edge Node pairs you deploy for PKS.

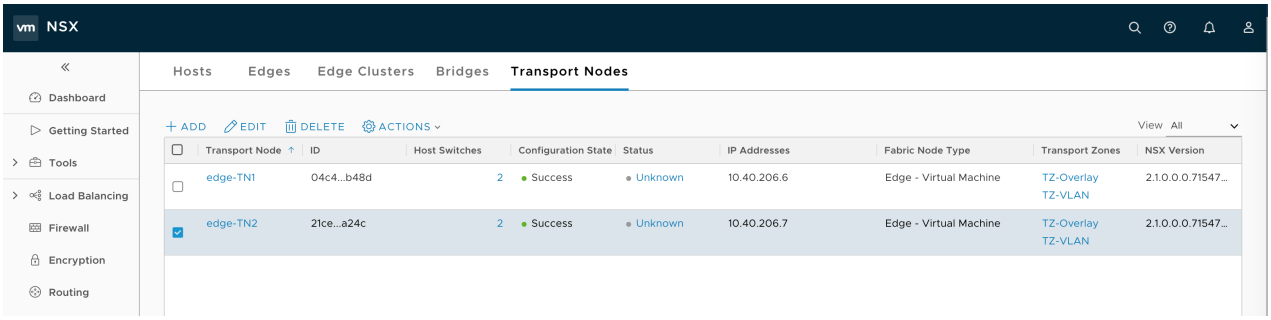
To verify the creation of Edge Transport Nodes:

1. In NSX Manager, select **Fabric > Nodes > Edges**.
2. Verify that Controller Connectivity and Manager Connectivity are `UP` for both Edge Nodes.



3. In NSX Manager, select **Fabric > Nodes > Transport Node**.

4. Verify that the configuration state is `Success`.



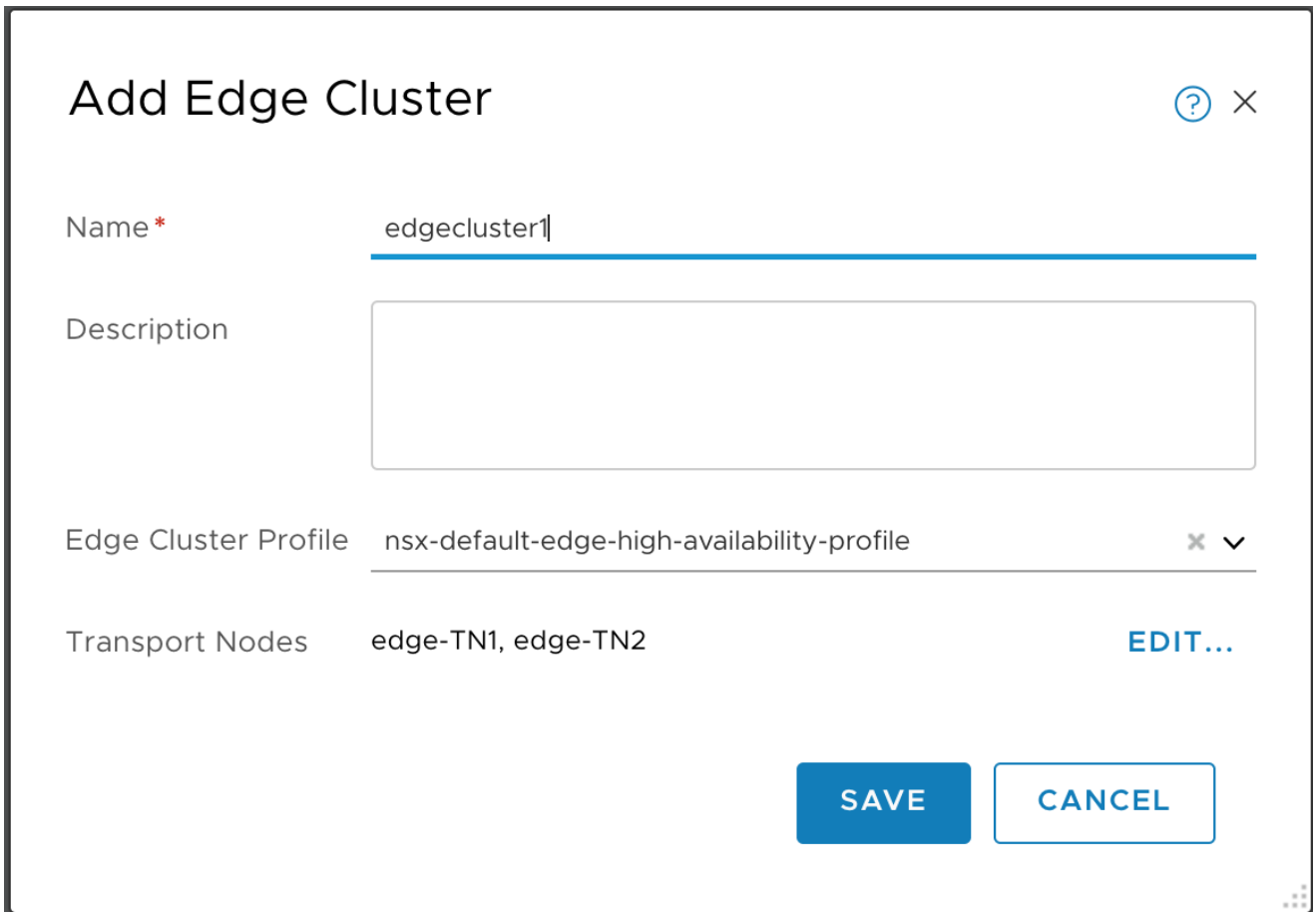
5. SSH to each NSX Edge VM and verify that the Edge Transport Node is “connected” to the Controller.

```
nsx-edge-1> get controllers
```

Step 12: Create Edge Cluster

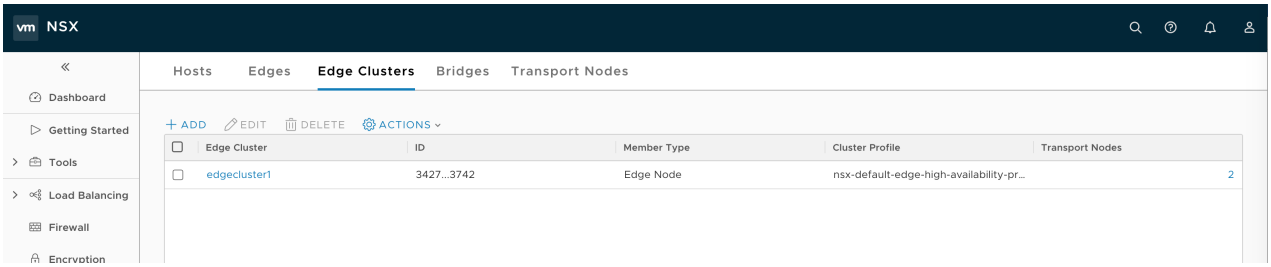
Create an NSX Edge Cluster and add each Edge Transport Node to the Edge Cluster by completing this procedure: Create an NSX Edge Cluster.

When creating the Edge Cluster for PKS, refer to the following example:

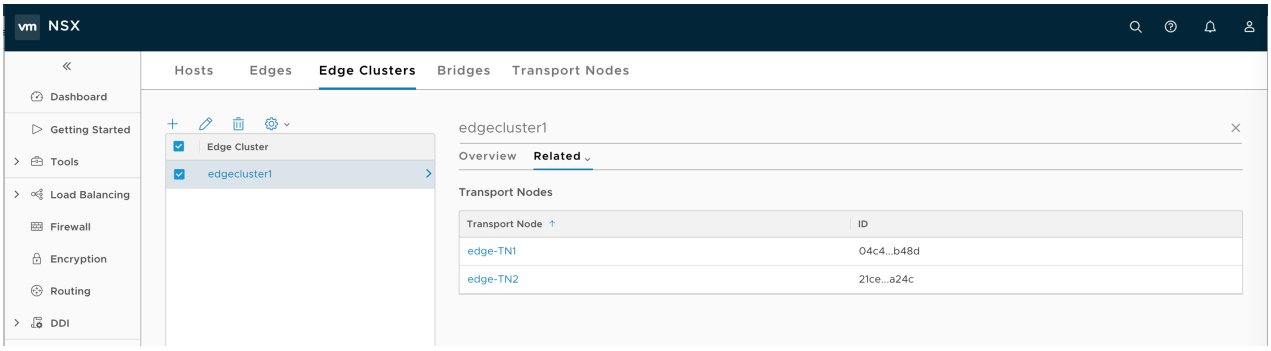


To verify Edge Cluster creation:

1. In NSX Manager, select **Fabric > Nodes > Edge Clusters**.
2. Verify that you see the new Edge Cluster.



3. Select **Edge Cluster > Related > Transport Nodes**.
4. Verify that all Edge Transport Nodes are members of the Edge Cluster.



5. SSH to NSX Edge Node 1 and run the following commands to verify proper connectivity.

```
nsx-edge-1> get vteps
nsx-edge-1> get host-switches
nsx-edge-1> get edge-cluster status
nsx-edge-1> get controller sessions
```

6. SSSH to NSX Edge Node 2 and repeat the above commands to verify proper connectivity.

7. Verify Edge-TN1 to Edge-TN2 connectivity (TEP to TEP).

```
nsx-edge-1> get logical-router
nsx-edge-1> vrf 0
nsx-edge-1(vrf)> ping IP-ADDRESS-EDGE-2
```

Step 13: Create T0 Logical Router

Create a Tier-0 Logical Router for PKS. The Tier-0 Logical Router is used to route data between the physical network and the NSX-T-defined virtual network.

To create a Tier-0 (T0) logical router:

1. Define a T0 logical switch with an ingress/egress uplink port. Attach the T0 LS to the VLAN Transport Zone.

2. Create a logical router port and assign to it a routable CIDR block, for example `10.172.1.0/28`, that your environment uses to route to all PKS assigned IP pools and IP blocks.

3. Connect the T0 router to the uplink VLAN logical switch.

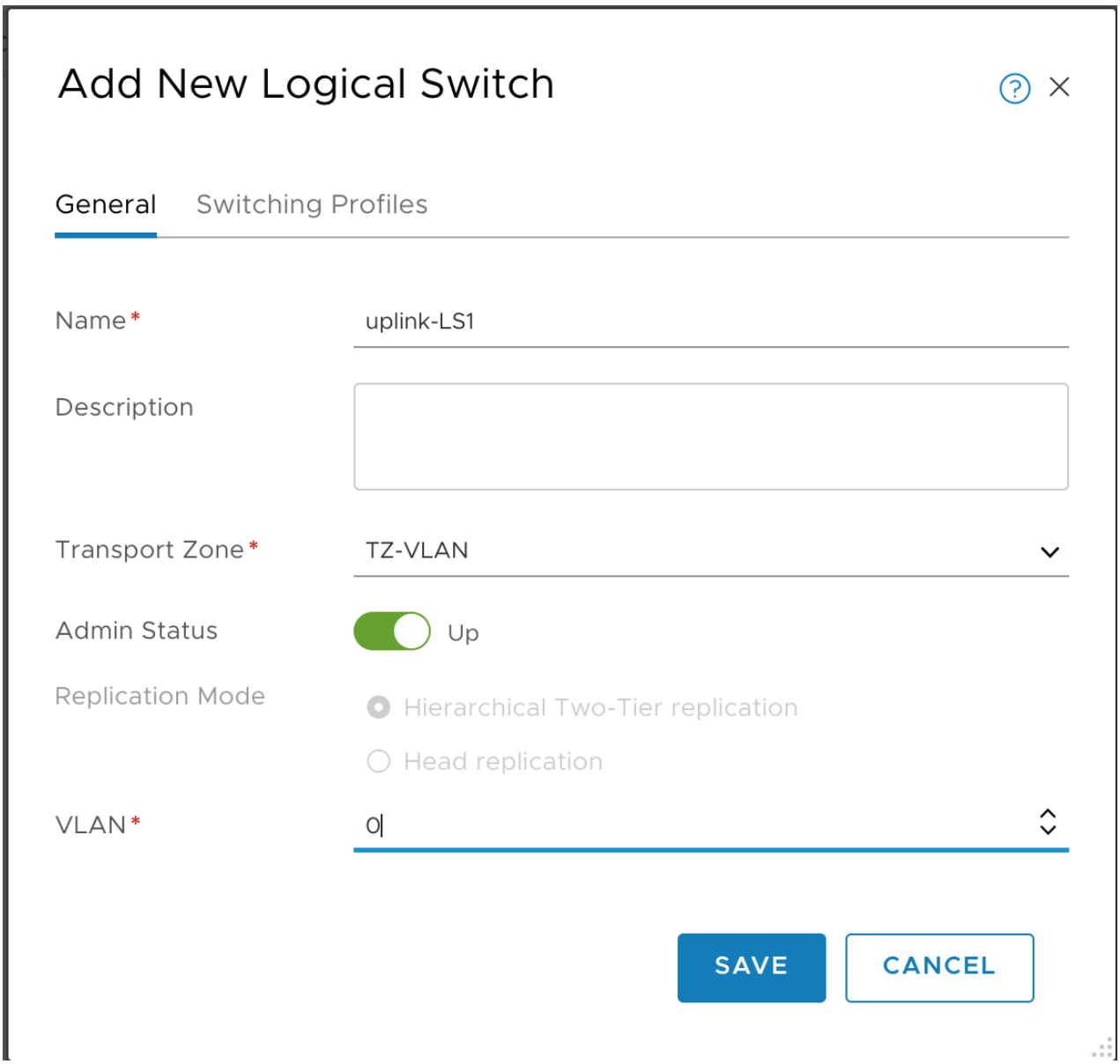
4. Attach the T0 router to the Edge Cluster and set HA mode to **Active-Standby**. NAT rules are applied on the T0 by NCP. If the T0 router is not set in **Active-Standby** mode, the router does not support NAT rule configuration.

5. Lastly, configure T0 routing to the rest of your environment using the appropriate routing protocol for your environment or by using static routes.

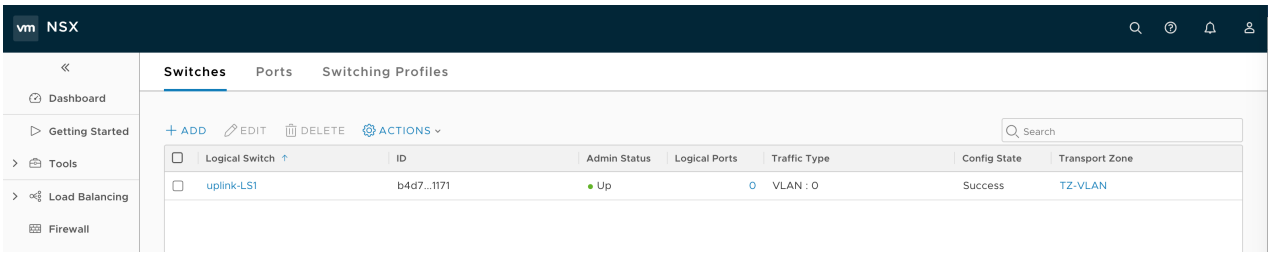
Create VLAN Logical Switch (LS)

1. In NSX Manager, go to **Switching > Switches**.

2. Click **Add** and create a VLAN logical switch (LS). For example:

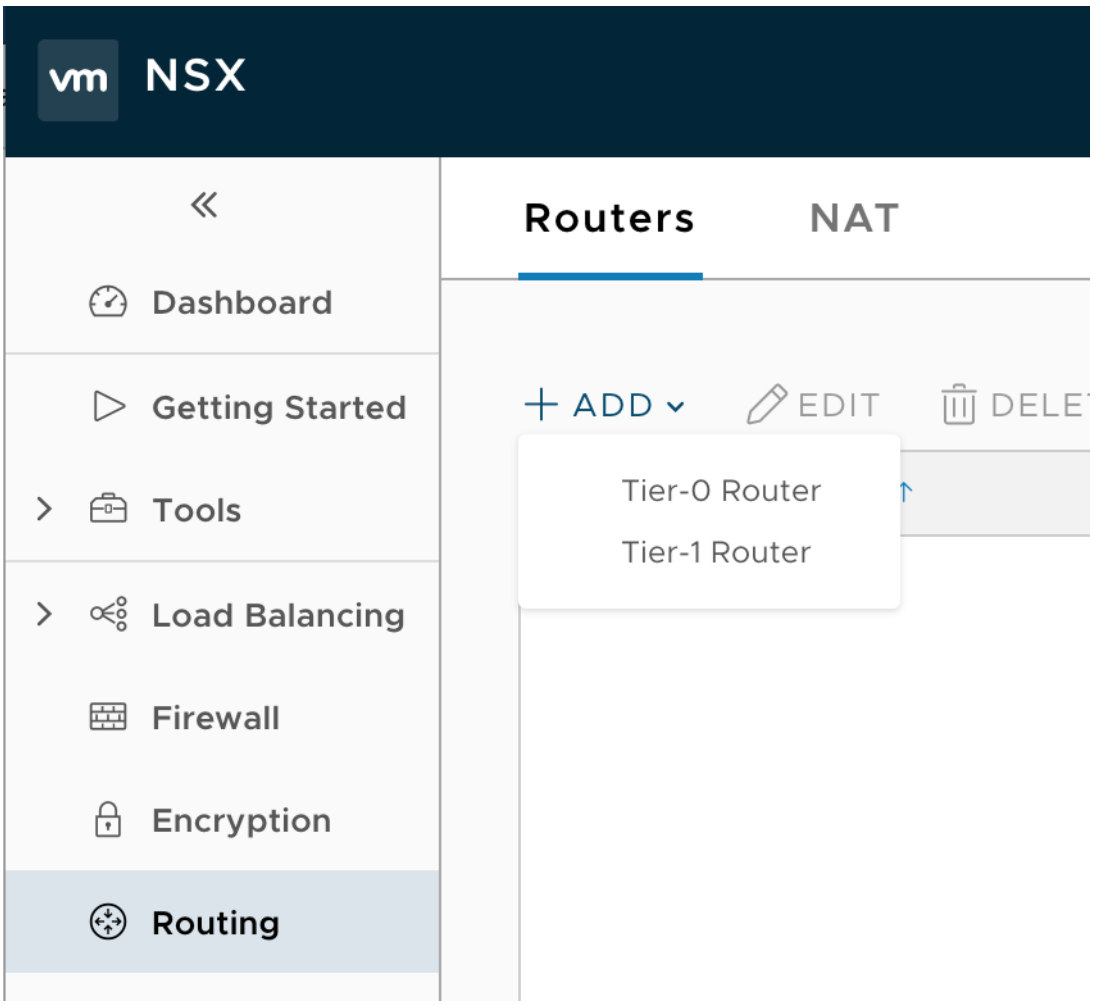


3. Click **Save** and verify that you see the new LS:



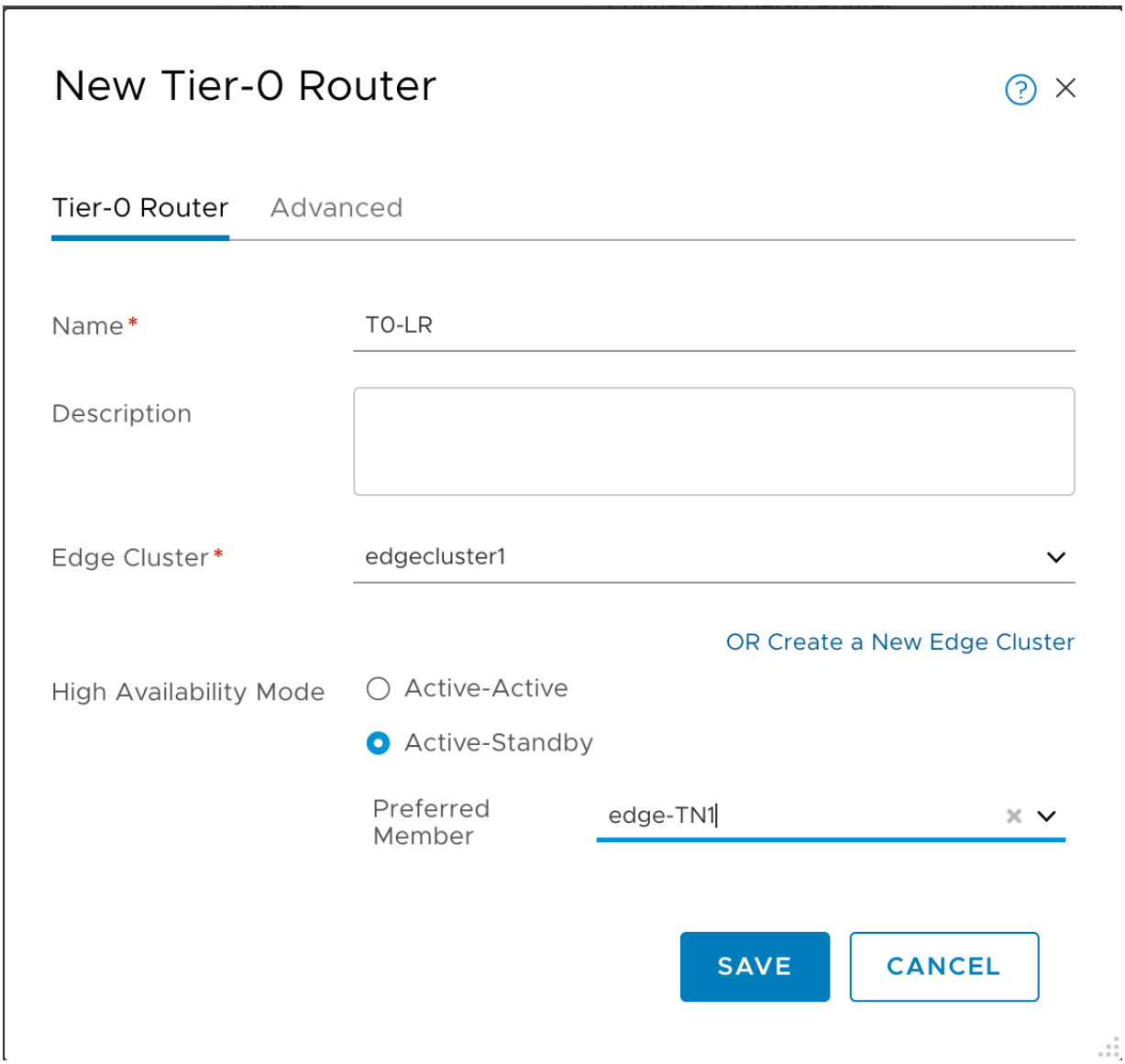
Create T0 Router Instance

1. In NSX Manager, go to **Routing > Routers**.
2. Click **Add** and select the **Tier-0 Router** option.

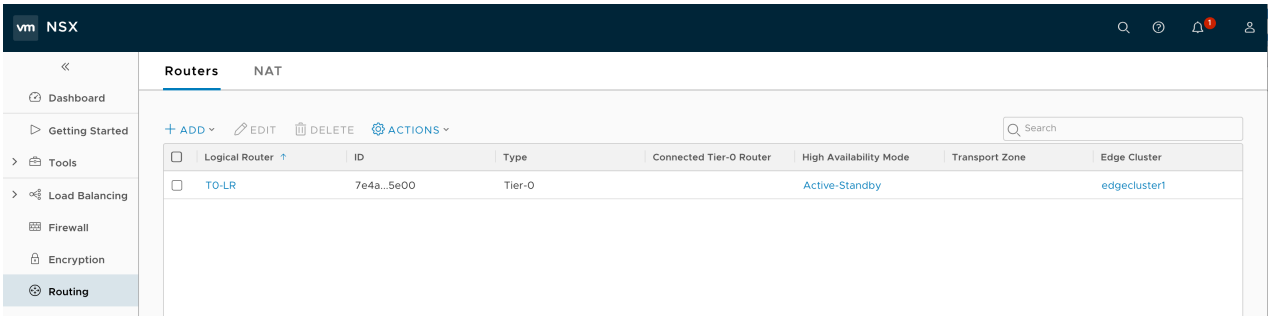


3. Create new T0 router as follows:

    - **Name**: Enter a name for the T0 router, such as `T0-LR` or `t0-pks`, for example.
    - **Edge Cluster**: Select the Edge Cluster, `edgecluster1` or `edge-cluster-pks`, for example.
    - **High Availability Mode**: Select `Active-Standby` (required).



4. Click **Save** and verify you see the new T0 Router instance:

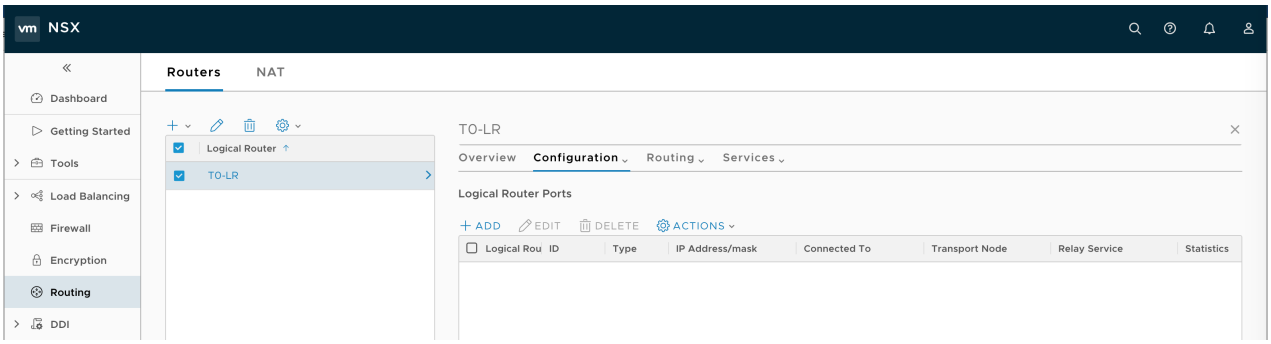


Note: Be sure to select Active/Standby. NAT rules are be applied on T0 by NCP. If not set Active-Standby, NCP will not be able to create NAT rules on the T0 Router.

Create T0 Router Port

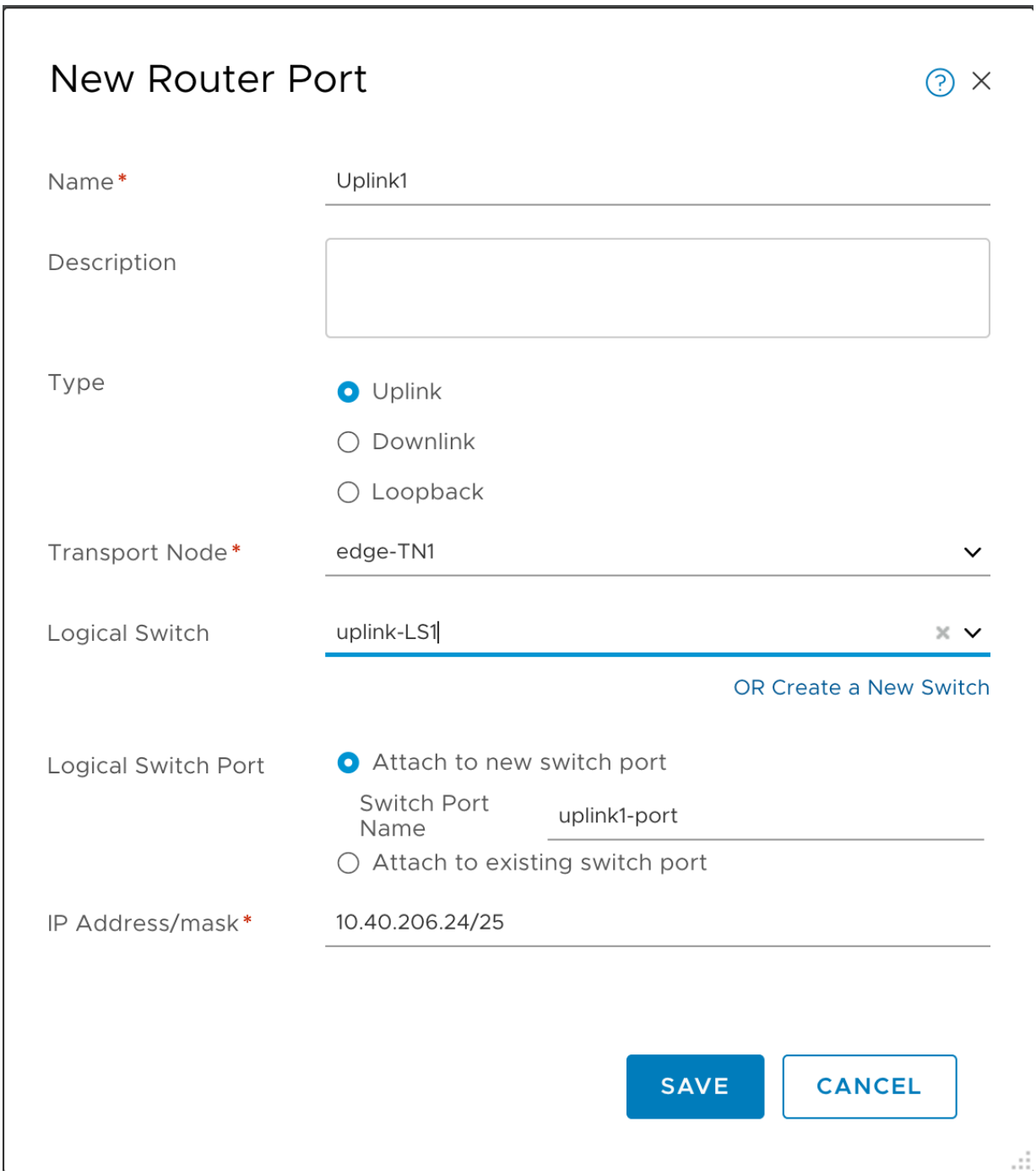
1. In NSX Manager, go to **Routing > Routers**.
2. Select the T0 Router you just created.
3. Select **Configuration > Router Ports**.

4. Select the T0 Router and click **Add**.

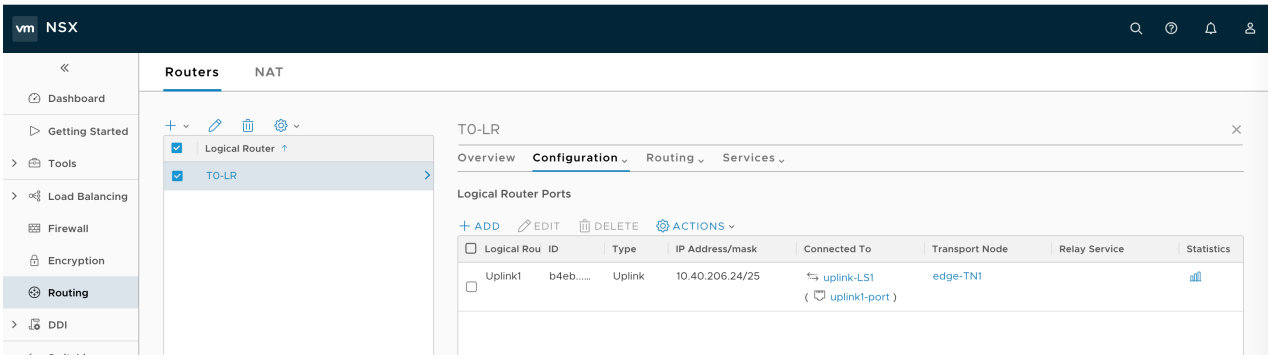


5. Create new T0 router port. Attach the T0 router port to the uplink logical switch you created (`uplink-LS1`, for example). Assign an IP address and CIDR that your environment uses to route to all PKS assigned IP pools and IP blocks. For example:

    - **Name**: `Uplink1`
    - **Type**: Uplink
    - **Transport Node**: `edge-TN1`
    - **Logical Switch**: `uplink-LS1`
    - **Logical Switch Port**: `uplink1-port`
    - **IP Address/mask**: `10.40.206.24/25` (for example)



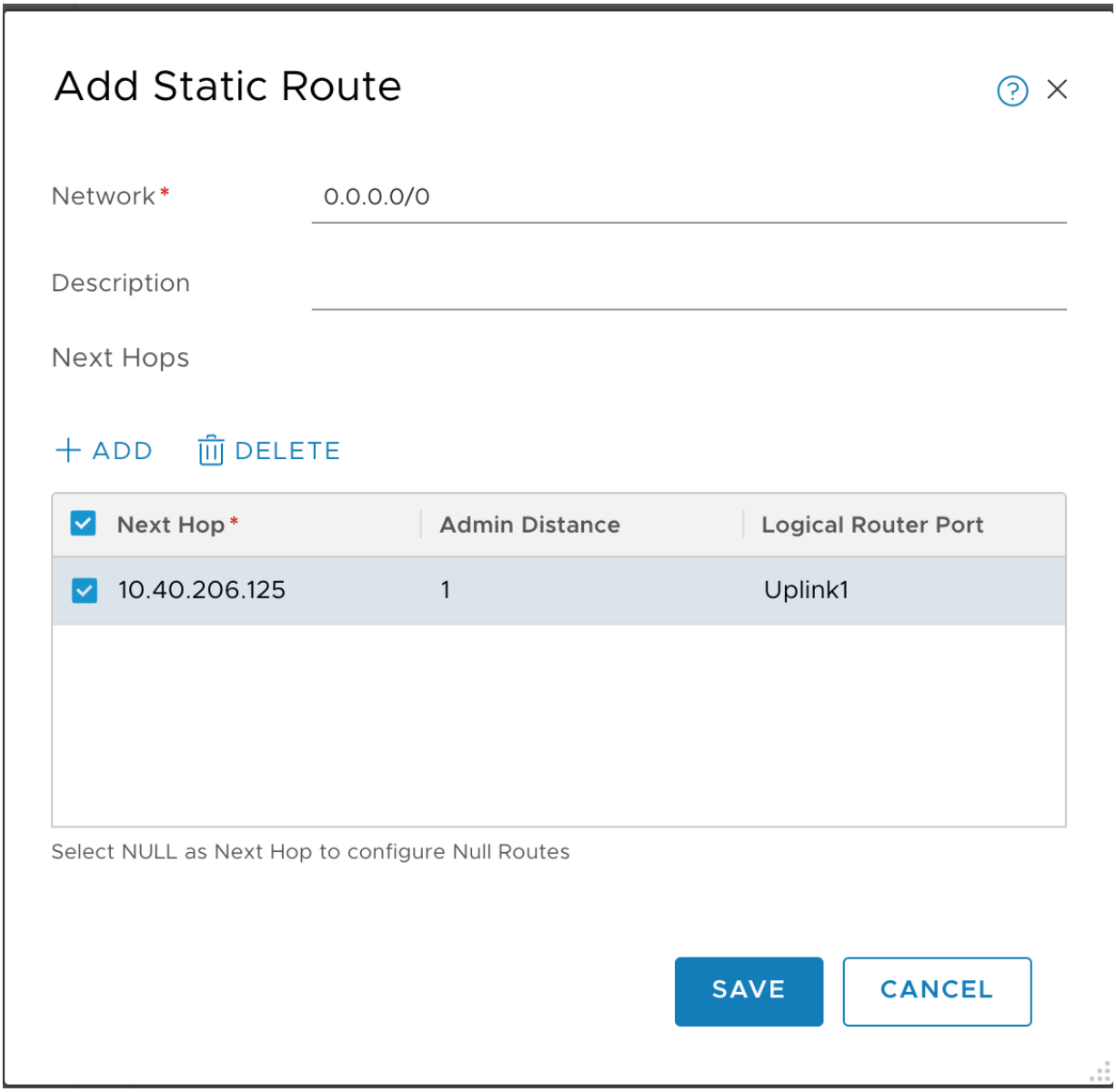
6. Click **Save** and verify that you see the new port interface:



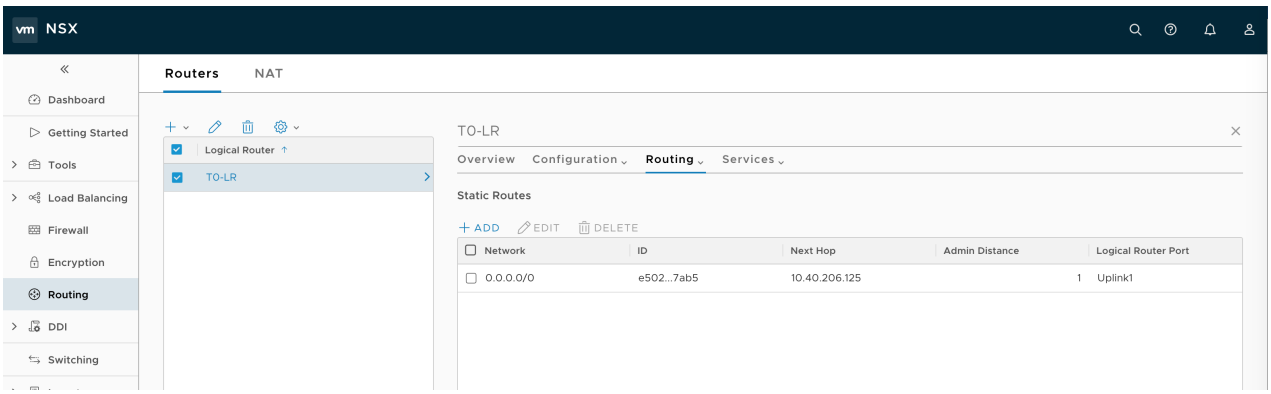
Define Default Static Route

Configure T0 routing to the rest of your environment using the appropriate routing protocol (if you are using no-NAT-mode), or using static routes (if you are using NAT-mode). The following example uses static routes for the T0 router. The CIDR used must route to the IP you just assigned to your T0 uplink interface.

1. Go to **Routing > Routers** and select the T0 Router.
2. Select **Routing > Static Routes** and click **Add**.
3. Create a new static route for the T0 router. For example:
    - **Network:** `0.0.0.0/0`
    - **Next Hop:** `10.40.206.125` (for example)
    - **Admin Distance:** `1`
    - **Logical Router Port:** `Uplink1`



4. Click **Save** and verify that see the newly created static route:



Verify T0 Router Creation

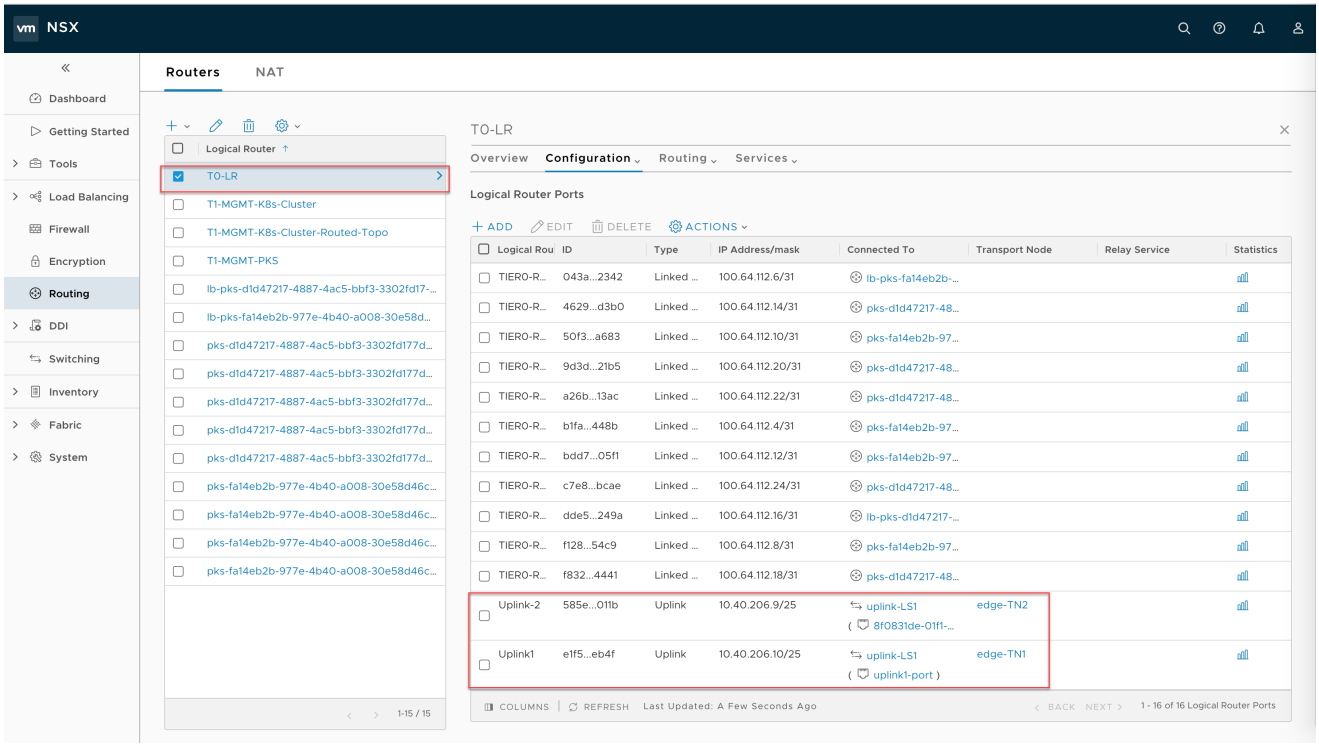
The T0 router uplink IP should be reachable from the corporate network. From your local laptop or workstation, ping the uplink IP address. For example:

```
PING 10.40.206.24 (10.40.206.24): 56 data bytes
64 bytes from 10.40.206.24: icmp_seq=0 ttl=53 time=33.738 ms
64 bytes from 10.40.206.24: icmp_seq=1 ttl=53 time=36.965 ms
```

Step 14: Configure Edge Nodes for HA

Configure high-availability (HA) for NSX Edge Nodes. If the T0 Router is not correctly configured for HA, failover to the standby Edge Node will not occur.

Proper configuration requires two new uplinks on the T0 router: one attached to Edge TN1, and the other attached to Edge TN2. In addition, you need to create a VIP that is the IP address used for the T0 uplink defined when the T0 Router was created.

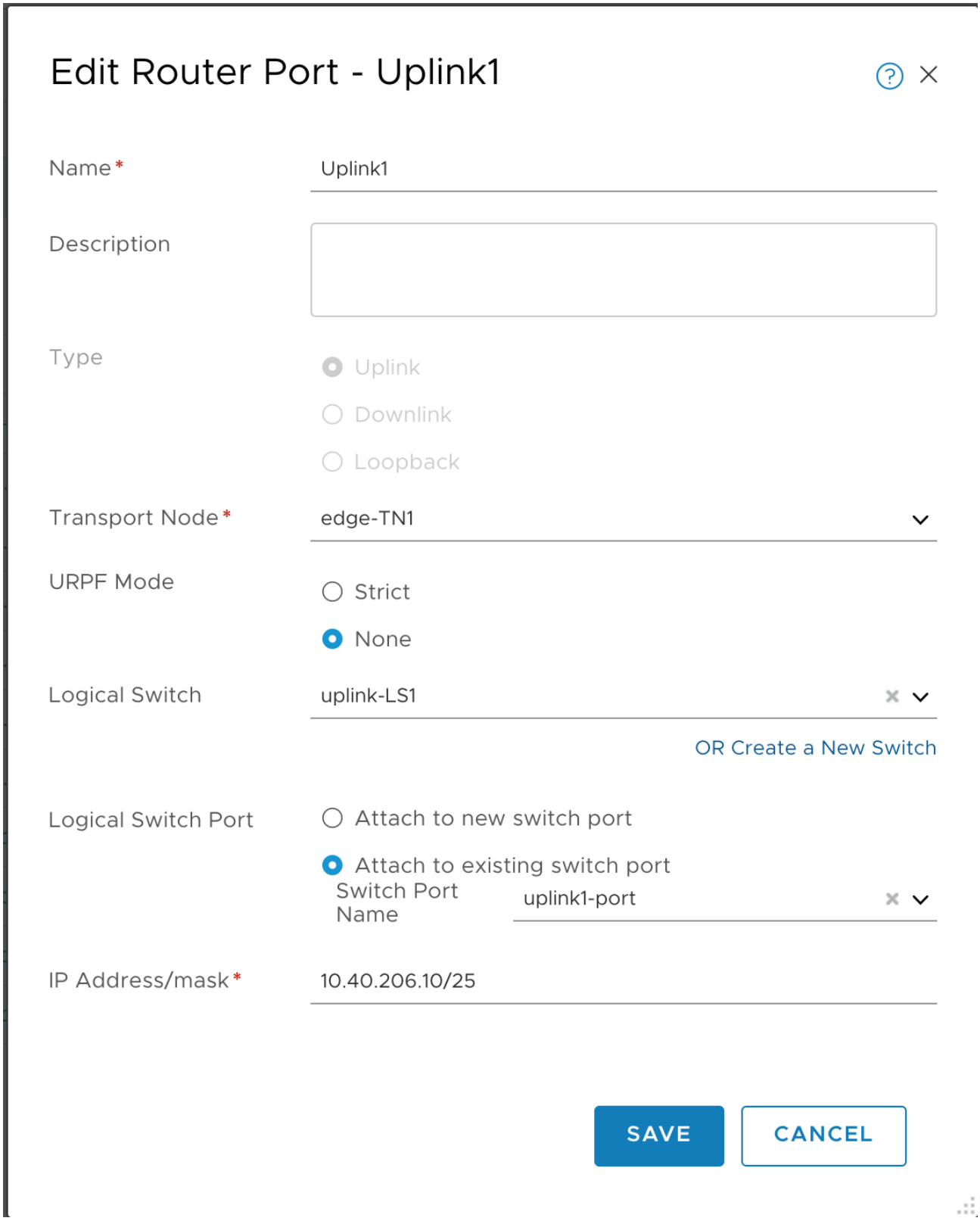


Create Uplink1 for Edge-TN1

On the T0 router, create the Uplink1 router port and attach it to Egde TN1. For example:

- **IP Address/Mask**: For example, `10.40.206.10/25`

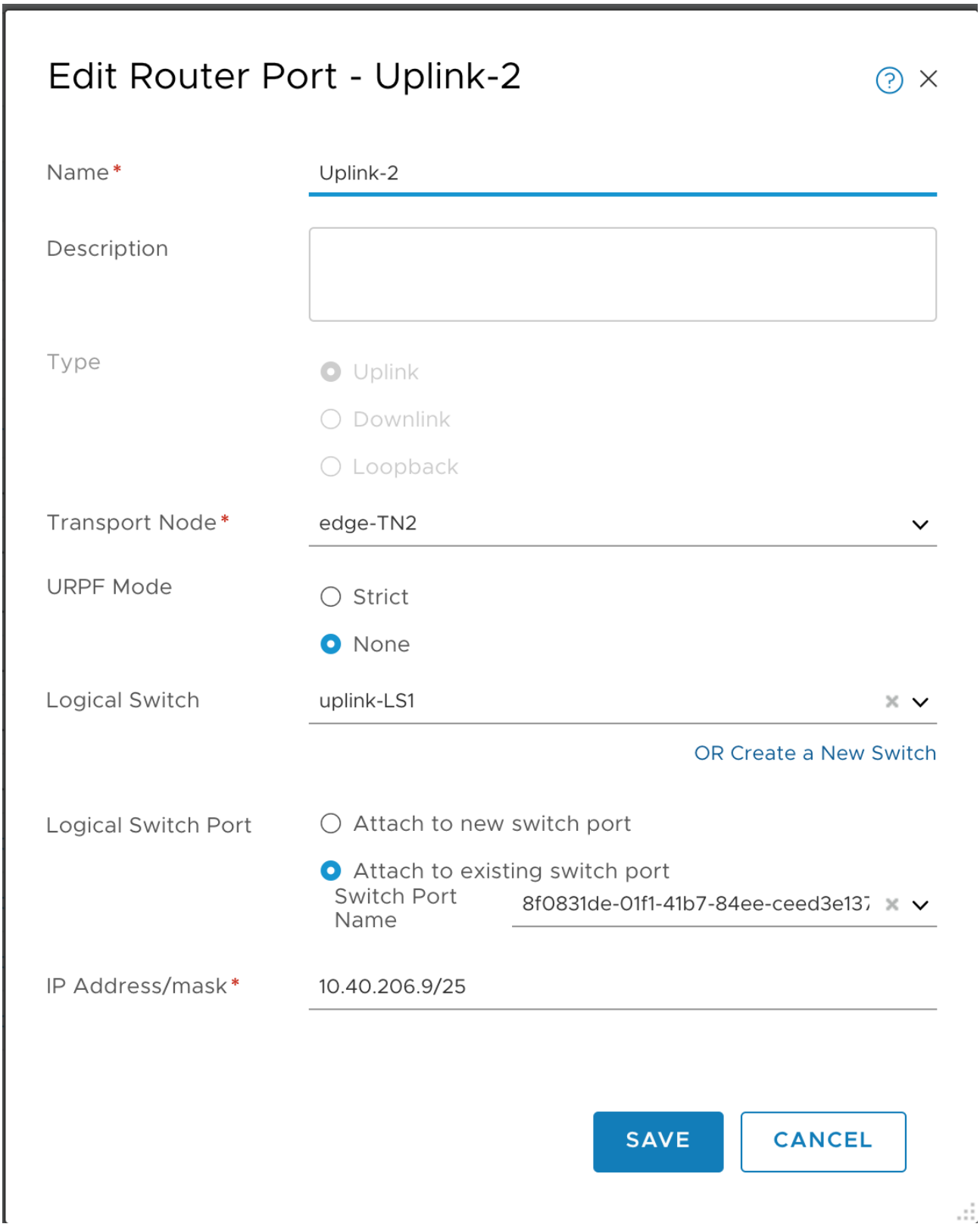
- **URPF Mode**: None (optional)
- **Transport Node**: `edge-TN1`
- **Logical Switch**: `uplink-LS1`



Create Uplink2 for Edge-TN2

On the T0 router, create the Uplink2 router port and attach it to Egde TN2. For example:

- **IP Address/Mask**: For example, `10.40.206.9/25`
- **URPF Mode**: None (optional)
- **Transport Node**: `edge-TN2`
- **Logical Switch**: `uplink-LS1`

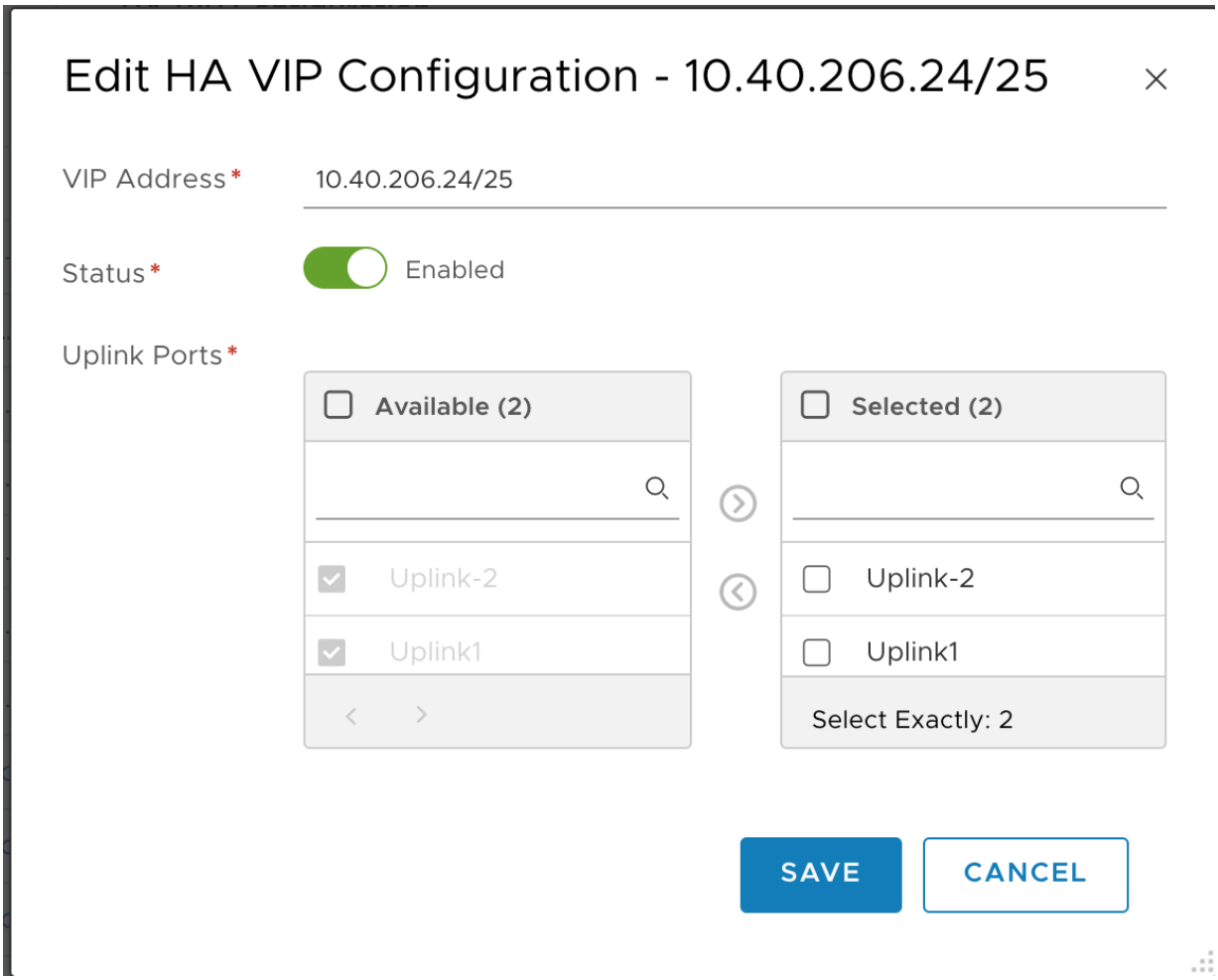


Create HA VIP

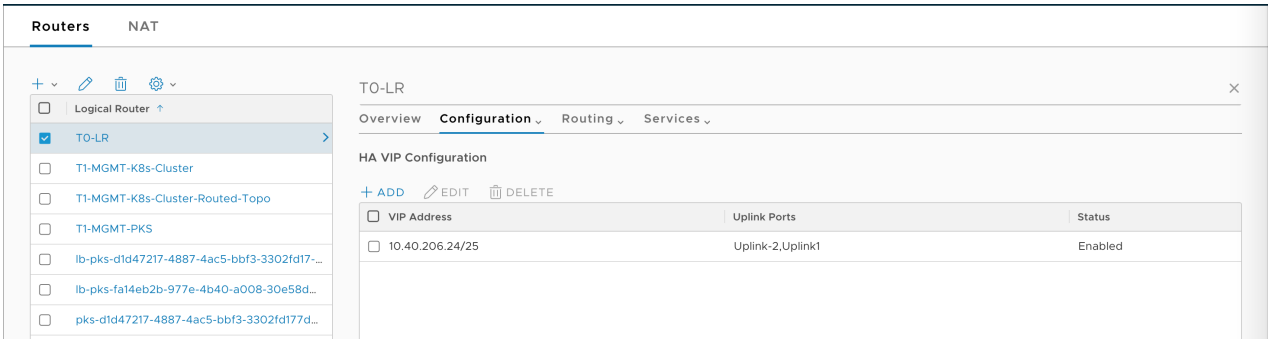
Create an HA virtual IP (VIP) address. This address is used for the T0 router uplink. External router devices, such as the physical router, peering with the T0 router must use this IP address.

Note: The IP addresses for uplink-1, uplink-2 and HA VIP must belong to same subnet.

1. On the T0 router, create the HA VIP. For example:
   - **VIP Address**: `10.40.206.24/25`
   - **Uplinks Ports**: `Uplink-1` and `Uplink-2`



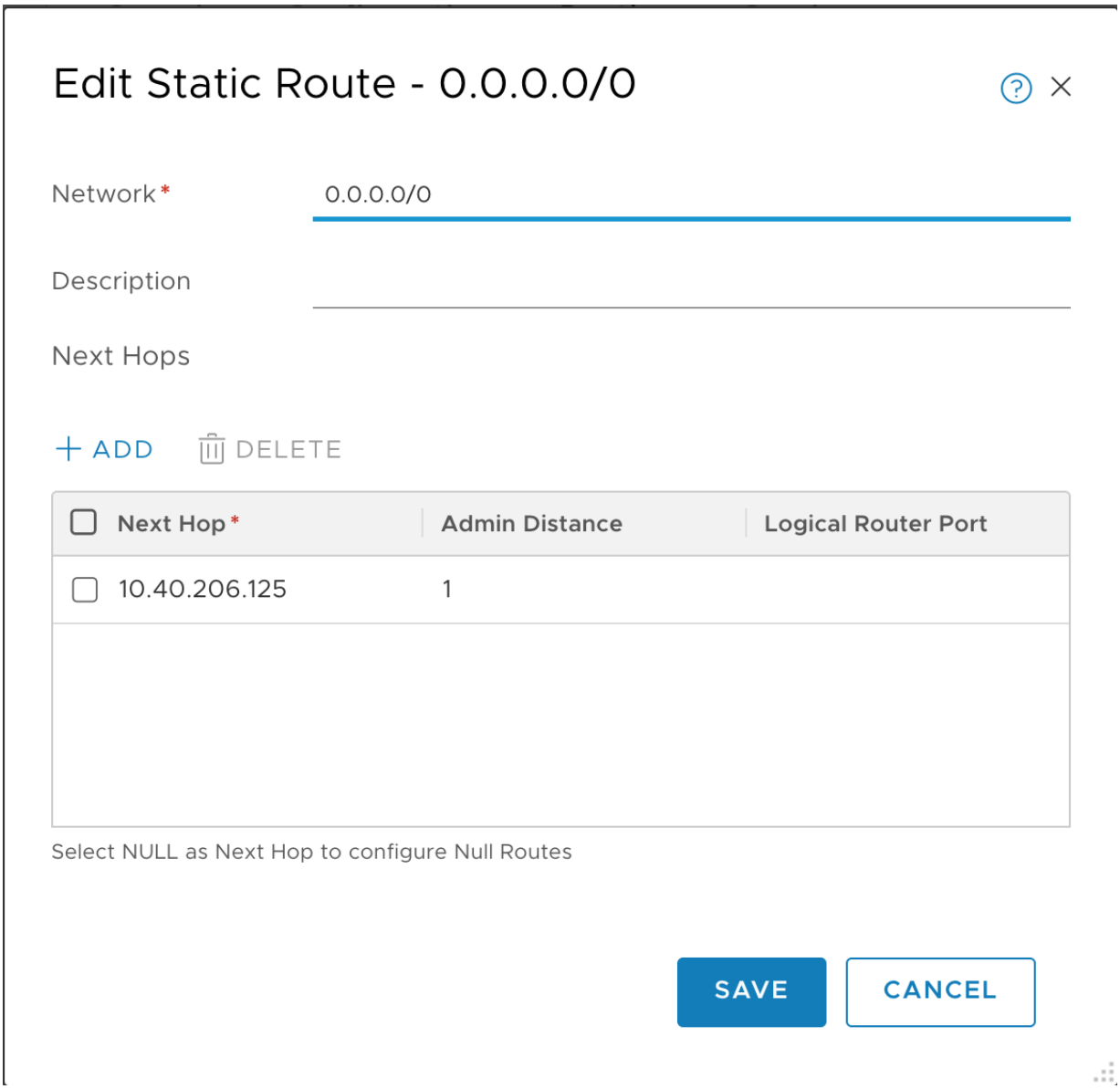
2. Verify creation of the HA VIP.



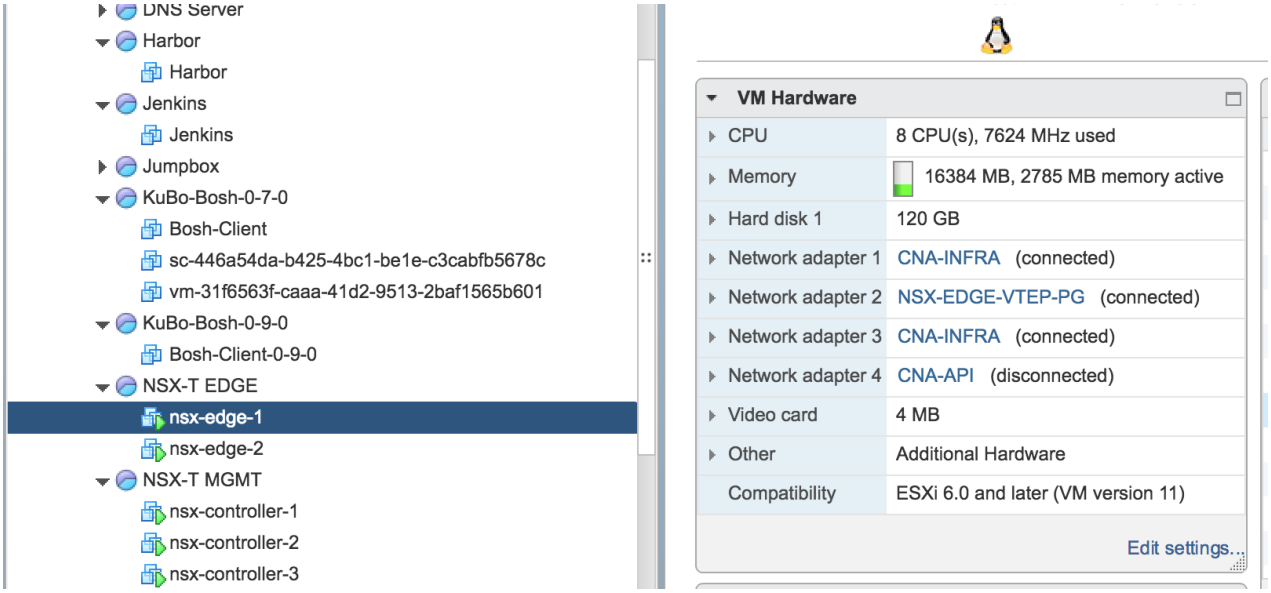
Create Static Route for HA

1. On the T0 router, create a static default route so that the next hop points to the physical router. For example:
   - **Network**: `0.0.0.0/0`
   - **Next Hop**: `10.40.206.125`

- **Logical Router Port**: empty



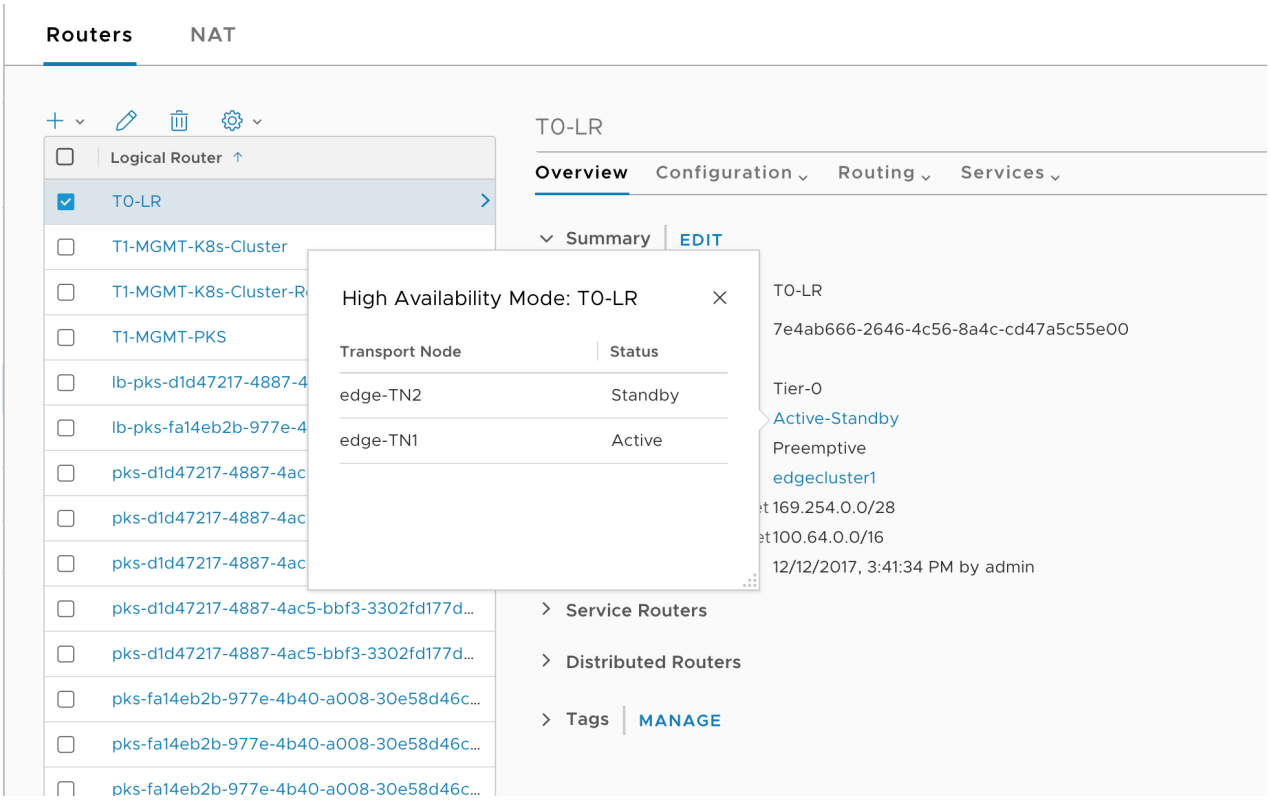
2. Using vCenter, disconnect any unused vNIC interface in each Edge Node VM (this interface can cause duplicate packets.) For example, in the screenshot below, **Network adpater 4** is not being used, so it is disconnected:



Note: Disconnect unused vNICs to prevent the duplication of traffic from two vNICs connected to same VLAN. This can occur when you configure HA for an active/standby Edge Node pair.

Verify Edge Node HA

1. The T0 router should display both Edge TNs in active/standby pairing.



2. Run the following commands to verify HA channels:

```
nsx-edge-n-1> get high-availability channels
nsx-edge-n-1> get high-availability channels stats
nsx-edge-n-1> get logical-router
nsx-edge-n-1> get logical-router ROUTER-UUID high-availability status
```

Step 15: Prepare ESXi Servers for the PKS Compute Cluster

For each ESXi host in the NSX-T Fabric to be used for PKS Compute purposes, create an associated transport node. For example, if you have three ESXi hosts in the NSX-T Fabric, create three nodes named `tnode-host-1`, `tnode-host-2`, and `tnode-host-3`. Add the Overlay Transport Zone to each ESXi Host Transport Node.

Prepare each ESXi server dedicated for the PKS Compute Cluster as a Transport Node. These instructions assume that for each participating ESXi host the ESXi hypervisor is installed and the `vmk0` is configured. In addition, each ESXi host must have at least one **free nic/vmnic** for use with NSX Host Transport Nodes that is not already in use by other vSwitches on the ESXi host. Make sure the `vmnic1` (second physical interface) of the ESXi host is not used. NSX will take ownership of it (opaque NSX vswitch will use it as uplink). For more information, see Add a Hypervisor Host to the NSX-T Fabric in the VMware NSX-T documentation.

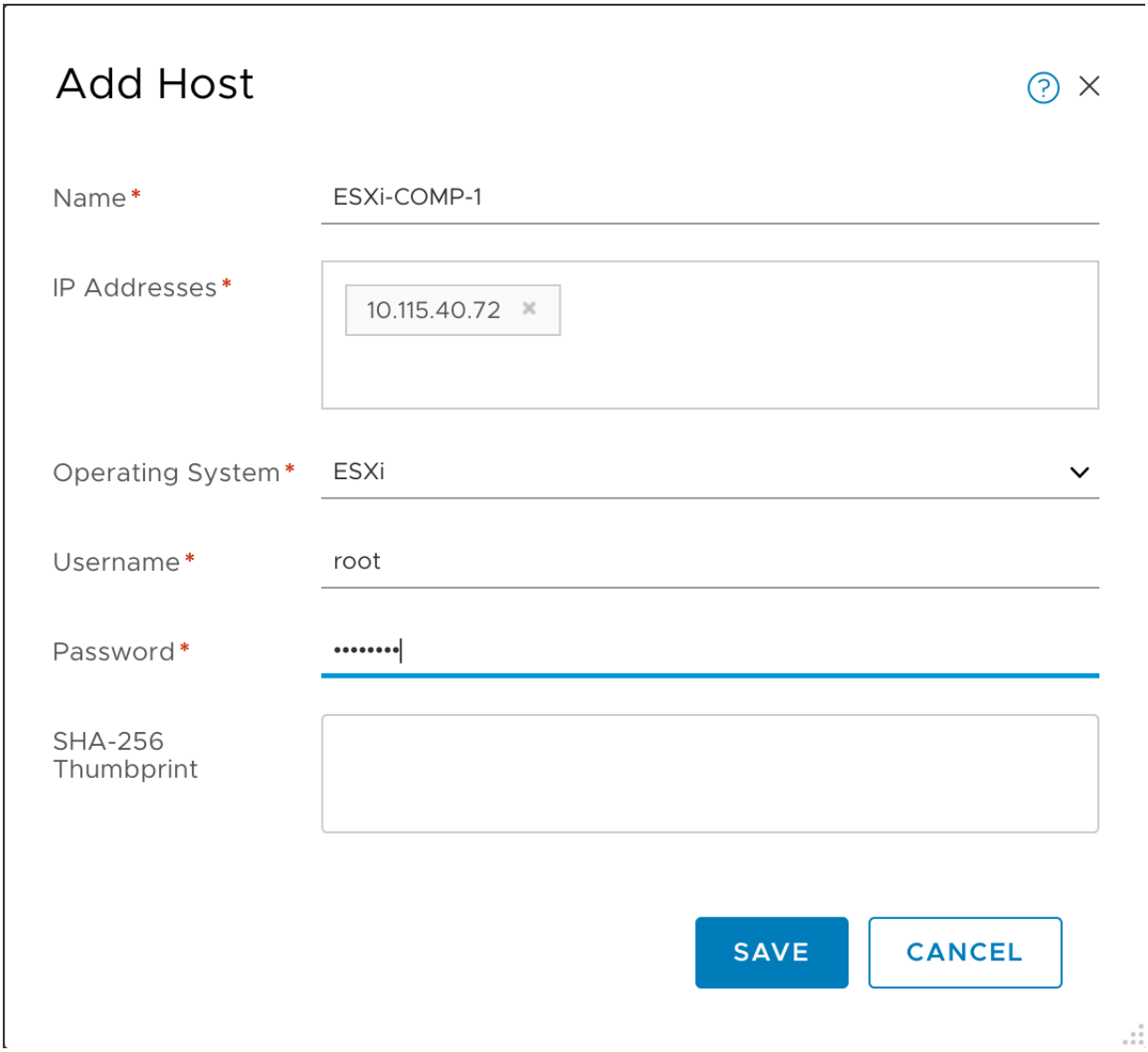
Add ESXi Host to NSX-T Fabric

Complete the following operation for each ESXi host to be used by the PKS Compute Cluster.

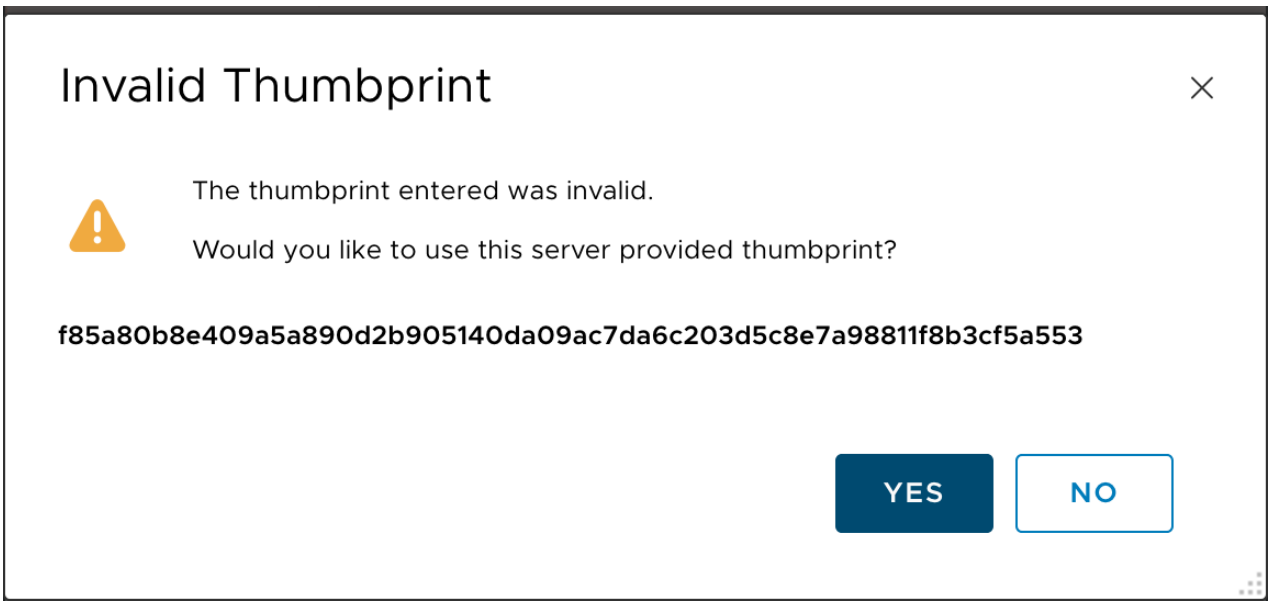
1. Go to **Fabric > Nodes > Hosts**.

2. Click **Add** and create a new host. For example:

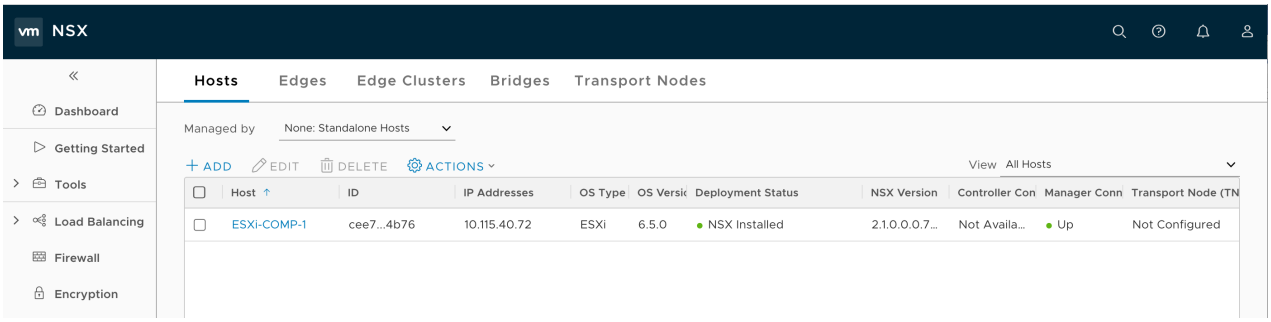
    - **IP Address**: 10.115.40.72
    - **OS**: ESXi
    - **Username**: root
    - **Password**: PASSWORD



3. After clicking **Save**, click **Yes** if the following invalid thumbprint message appears.

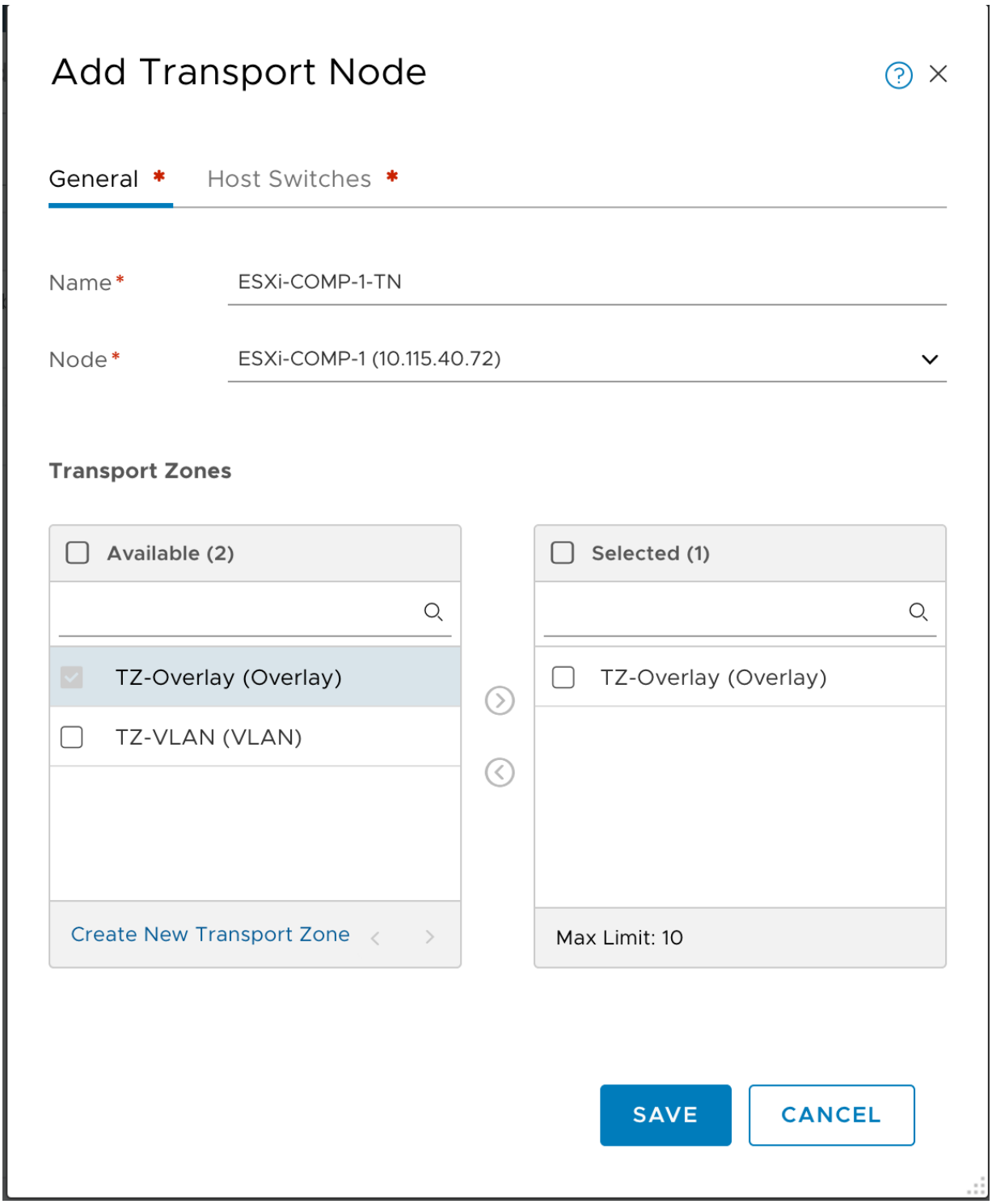


4. NSX installs VIBs on the ESXi host. In a few moments, you should see the new defined host. Deployment status should show `NSX Installed` and Manager Connectivity should show `Up`.



Create Transport Node

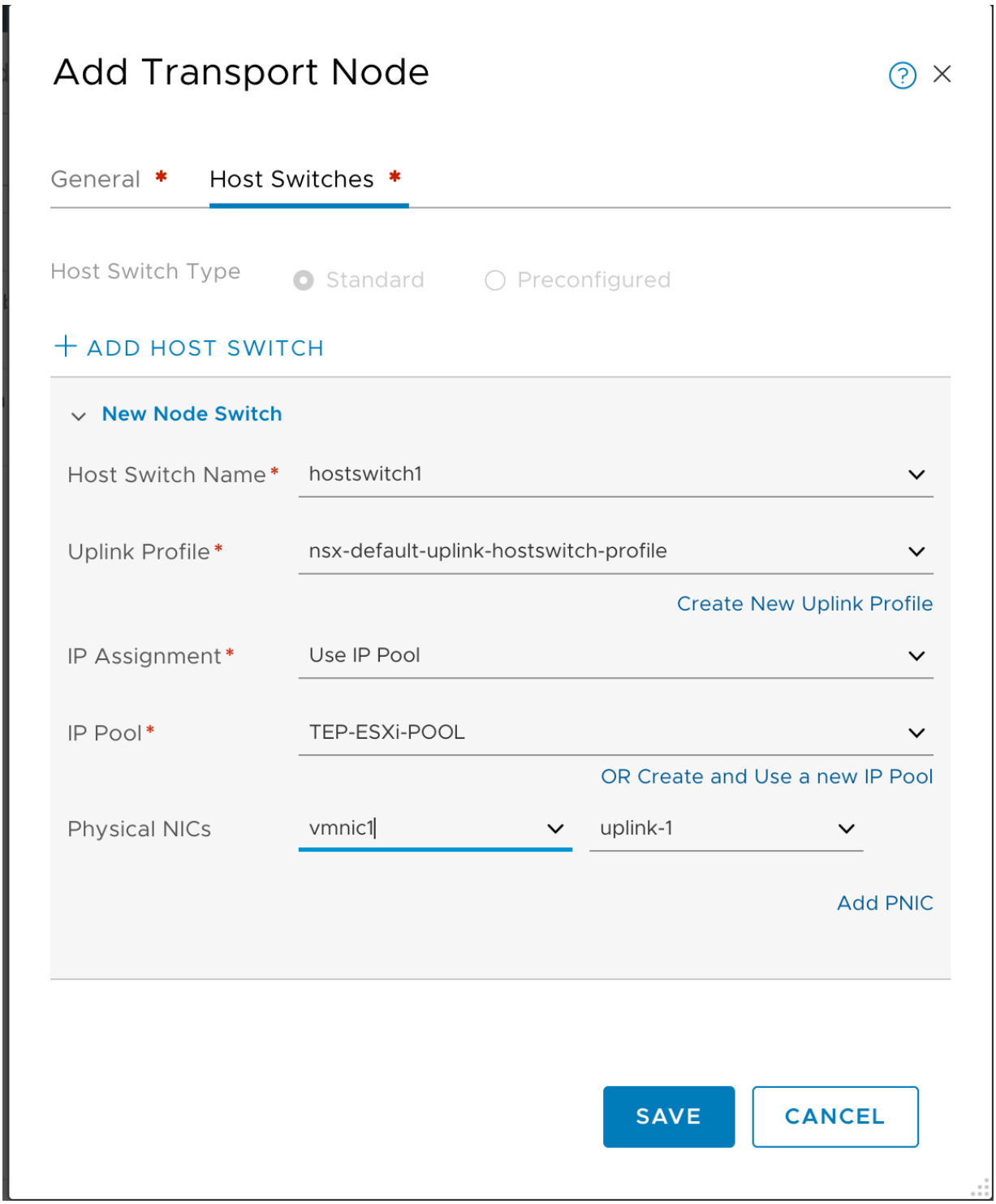
1. In NSX Manager, go to **Fabric > Nodes > Transport Nodes**.
2. Click **Add** and create a new Transport Node. For example:
   - **Name**: ESXi-COMP-1-TN
   - **Node**: ESXi-COMP-1
   - **TZ**: TZ-Overlay



3. Select the **Host Switches** tab.

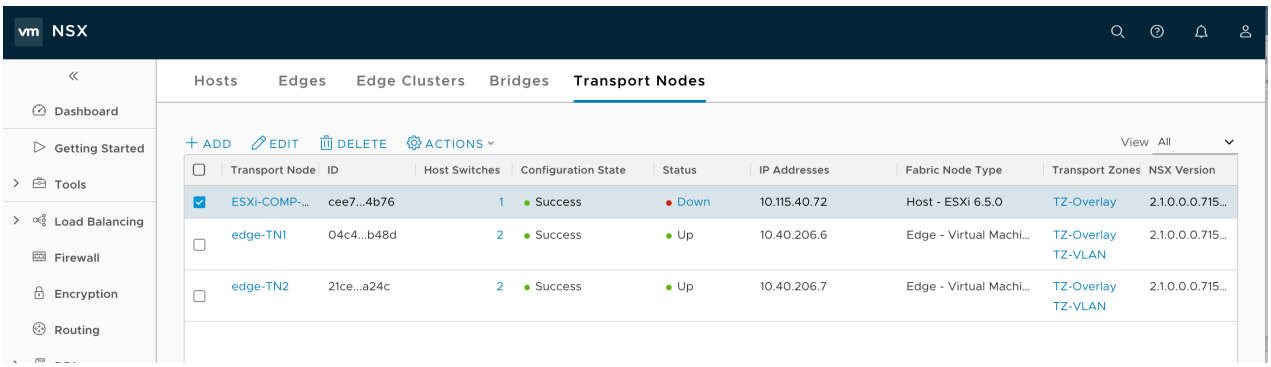
4. Configure a Host Switch. For example:

    - **Host Switch Name:** `hostswitch1`
    - **Uplink Profile:** `nsx-default-uplink-hostswitch-profile`
    - **IP Assignment:** `Use IP Pool`
    - **IP POOL:** `TEP-ESXi-POOL`
    - **Physical NICs:** `vmnic1`

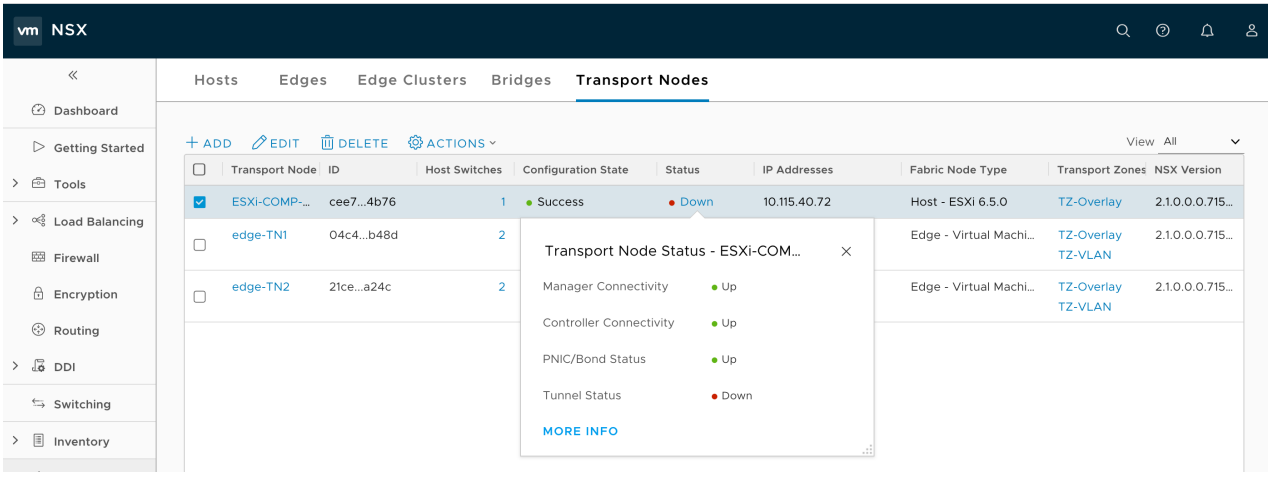


Verify ESXi Host Preparation for PKS Compute Cluster

1. Verify that you see the ESXi Compute Transport Node:



2. Verify the status is `Up`.



Note: If you are using NSX-T 2.3, the status should be up. If you are using NSX-T 2.2, the status may incorrectly show as down (because the Tunnel Status is Down.) Either way, verify TEP communications as described in the next step.

3. Make sure the NSX TEP vmk is created on ESXi host and TEP to TEP communication (with Edge TN for instance) works.

```
[root@ESXi-1:~] esxcfg-vmknic -l

[root@ESXi-1:~] vmkping ++netstack=vxlan <IP of the vmk10 interface> -d -s 1500
```

Next Step

After you complete this procedure, follow the instructions in Creating the PKS Management Plane.

Please send any feedback you have to pks-feedback@pivotal.io.

Deploying NSX-T v2.4 for Enterprise PKS

Page last updated:

To deploy NSX-T for Pivotal Container Service (PKS), complete the following set of procedures, in the order presented.

Note: The instructions provided in this topic are for NSX-T v2.4. If you are using NSX-T v2.3.1, see Deploying NSX-T v2.3.1 for Enterprise PKS.

Prerequisites

Before you begin this procedure, ensure that you have successfully completed all preceding steps for installing PKS on vSphere with NSX-T, including:

- vSphere with NSX-T Version Requirements
- Hardware Requirements for PKS on vSphere with NSX-T
- NSX-T Deployment Topologies for PKS
- Preparing to Deploy PKS with NSX-T on vSphere

NSX-T v2.4 Management Interfaces

This section describes the NSX-T v2.4 management interface options, differences, use cases, and recommendations.

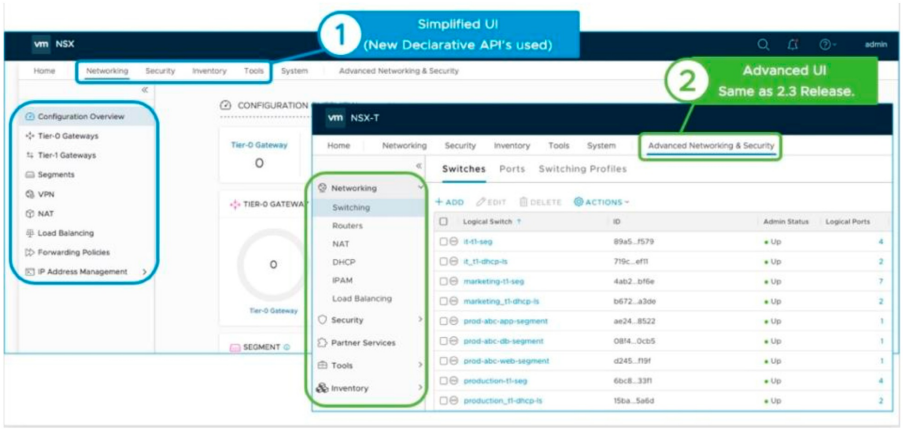
Note: NSX-T v2.4 implements a new Policy API and a new NSX Manager user interface (UI) based on the Policy API. Enterprise PKS does not support the Policy API or Policy-based UI. Enterprise PKS supports the NSX Management API, which is exposed via the “Advanced Networking” tab of the of the NSX Manager UI. When installing and configuring NSX-T v2.4 for use with Enterprise PKS, use the “Advanced Networking” tab to create any required networking objects.

Interface Options

With NSX-T 2.4 you have two options to interact with NSX Manager:

1. Simplified UI/API
    - New declarative interface introduced in NSX-T 2.4 that uses the new Declarative API/Data Model (Policy API).
    - The NSX-T Container Plugin (NCP) that is embedded in the Enterprise PKS tile does not support the Policy API at this time.
    - You cannot use the Simplified UI/API to manage NSX-T for use with Enterprise PKS upgrades and new installations.
2. Advanced UI/API
    - Legacy imperative interface based on the NSX Management API.
    - Provides the NSX-T v2.3 user interface to address Enterprise PKS installation and upgrade use cases. Currently NCP only supports the Management API.
    - The Advanced UI/API will be deprecated over time; all features and use cases will eventually be transfered to the Simplified UI/API.

As shown in the picture below, for all Enterprise PKS workloads, use the **Advanced Networking and Security** tab to create, read, update, and delete required network objects. For NSX-T host perparation and configuration, such as deploying NSX Managers and Edge Nodes, use the **System** tab. Do not use the “Simplified UI” for Enterprise PKS objects.



Note: The NSX-T Container Plugin (NCP) that is embedded in the Enterprise PKS tile does not currently support the Policy API. Make sure you use the **Advanced Networking and Security** tab of the of the NSX Manager UI when configuring NSX-T for use with Enterprise PKS.

Upgrading to Enterprise PKS v1.4 and NSX-T v2.4

In the case of upgrade from NSX-T v2.3 to v2.4, the existing NSX-T v2.3 configuration is copied to NSX-T v2.4 under the **Advanced Networking and Security** tab. The network objects required by PKS can only be managed from this user interface. In other words, this configuration will not be shown in the Simplified UI. When you upgrade to NSX-T v2.4, the Simplified UI will show a information banner that indicates the objects are available in the “Advanced Networking” tab.

For instructions on upgrading NSX-T from v2.3 to v2.4 for Enterprise PKS, see Upgrading Enterprise PKS with NSX-T.

Installing Enterprise PKS v1.4 with NSX-T v2.4

To perform a new installation of NSX-T v2.4 with Enterprise PKS v1.4, complete the steps.

1. Prepare for Installing NSX-T 2.4.
2. Install NSX Manager using the **System** tab.
3. Deploy Additional NSX Manager Nodes to Form a Management Cluster using the **System** tab.
4. Assign a Virtual IP Address and Certificate to the NSX-T Manager Cluster using the **System** tab.
5. Install One or More Pairs of NSX Edge Nodes using the **System** tab.

   Warning: For Enterprise PKS you must install a large size VM form factor or the bare metal Edge Node. See Deploy NSX Edge Nodes for more information.
6. Create an NSX Edge Cluster using the **System** tab.
7. Join NSX Edge Nodes with the Management Plane using the **System** tab.
8. Enable Repository Service on NSX Manager. Repeat this step for each NSX Manager.
9. Create an IP Pool for Tunnel Endpoint IP Addresses using the **System** tab.
10. Create Overlay and VLAN Transport Zones using the **System** tab.
11. Create an Uplink Profile using the **System** tab.
12. Create Edge Transport Nodes using the **System** tab.
13. Configure Edge Nodes for HA using the **System** tab.
14. Prepare ESXi Hosts as Transport Nodes for NSX-T using the **System** tab.

15. Create a Tier-0 Logical Router using the **Advanced Networking and Security** tab in NSX Manager.
16. Create the Enterprise PKS Management Plane using the **Advanced Networking and Security** tab in NSX Manager.
17. Create the PKS Compute Plane using the **Advanced Networking and Security** tab in NSX Manager.
18. Deploy Ops Manager 2.4.3+.
19. Generate and Register the NSX Manager Cluster Certificate (if you have not already done so).
20. Configure BOSH Director with NSX-T for Enterprise PKS.
21. Generate and Register the NSX Manager Superuser Principal Identity Certificate and Key.
22. Create NSX-T Objects for Enterprise PKS using the **System** tab in NSX Manager.
23. Install Enterprise PKS on vSphere with NSX-T v2.4.

Please send any feedback you have to pks-feedback@pivotal.io.

Creating the PKS Management Plane

Page last updated:

Prepare the vSphere and NSX-T infrastructure for the PKS Management Plane where the PKS, Ops Manager, BOSH Director, and Harbor Registry VMs are deployed.

Prerequisites

Before you begin this procedure, ensure that you have reviewed the following documentation for installing PKS on vSphere with NSX-T:

- vSphere with NSX-T Version Requirements
- Hardware Requirements for PKS on vSphere with NSX-T
- NSX-T Deployment Topologies for PKS
- Preparing to Deploy PKS with NSX-T on vSphere

In addition, ensure that you have successfully deployed NSX-T for PKS. For more information, see Deploying NSX-T for PKS.

About the PKS Management Plane

The PKS Management Plane is the network for PKS Management components, including PKS, Ops Manager, and BOSH Director. The PKS Management Plane includes a vSphere resource pool for Management Plane components, as well as a NSX Tier-1 Logical Switch, Tier-1 Logical Router, and Router Port, as well as NSX NAT rules.

If you are using either the the NAT deployment topology or the No-NAT with Logical Switch deployment topology, create a Tier-1 (T1) Logical Switch, and a Tier-1 Logical Router and Port. Link the T1 logical router to the T0 logical router, and select the Edge Cluster defined for PKS. Enable route advertisement for the T1 Logical Router and advertise All NSX connected routes for the PKS Management Plane VMs (PKS, Ops Manager, and BOSH Director).

If you are using the NAT Topology, create the following NAT rules on the T0 Router.

- Destination NAT (DNAT) rule that maps an external IP address from the **PKS MANAGEMENT CIDR** to the IP where you deploy Ops Manager on the PKS Management logical switch. For example, a DNAT rule that maps `10.172.1.2` to `172.31.0.2`, where `172.31.0.2` is the IP address you assign to Ops Manager when connected to `ls-pks-mgmt`.
- (Optional) Destination NAT (DNAT) rule that maps an external IP address from the **PKS MANAGEMENT CIDR** to the IP where you deploy Harbor on the PKS Management logical switch. For example, a DNAT rule that maps `10.172.1.3` to `172.31.0.3`, where `172.31.0.3` is the IP address you assign to Harbor when connected to `ls-pks-mgmt`.
- Source NAT (SNAT) rule to allow the PKS Management VMs to communicate with your vCenter and NSX Manager environments. For example, an SNAT rule that maps `172.31.0.0/24` to `10.172.1.1`, where `10.172.1.1` is a routable IP address from your **PKS MANAGEMENT CIDR**.
- SNAT rule for PKS management components to access ESXi Hosts.
- (Optional) SNAT rules for access to other management servers, such as DNS, NTP, and LDAP/AD.

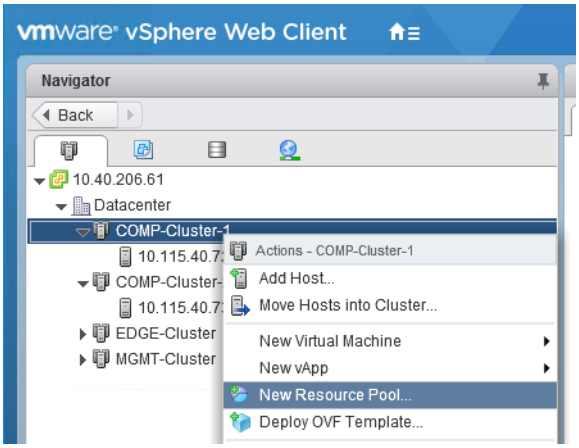
Lastly, for both NAT and no-NAT mode, if you want developers to be able to access the PKS API (that is, use the PKS CLI) from their workstations or laptops, you must share the PKS API endpoint to allow your organization to use the API to create, update, and delete clusters. For more information, see Creating Clusters.

Developers should use the DNAT IP address when logging in with the PKS CLI. For more information, see Using PKS. To create this DNAT rule, see Create DNAT Rule on T0 Router for External Access to the PKS CLI.

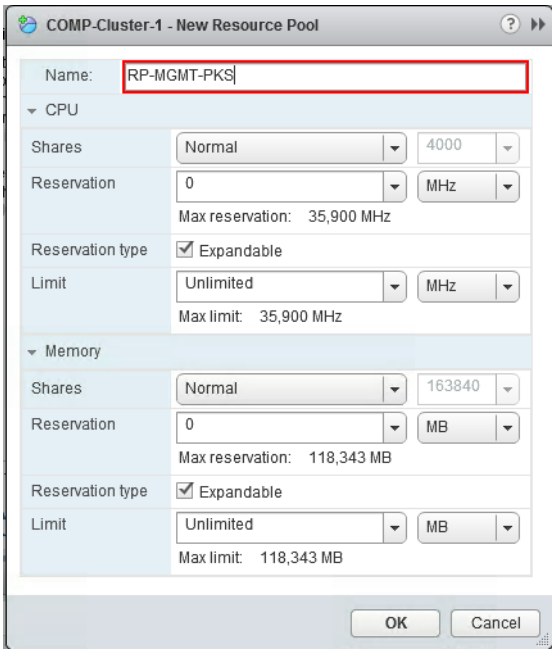
Step 1. Create vSphere Resource Pool for the PKS Management Plane

1. Log in to vCenter for your vSphere environment.

2. Select **Compute Cluster > New Resource Pool**.

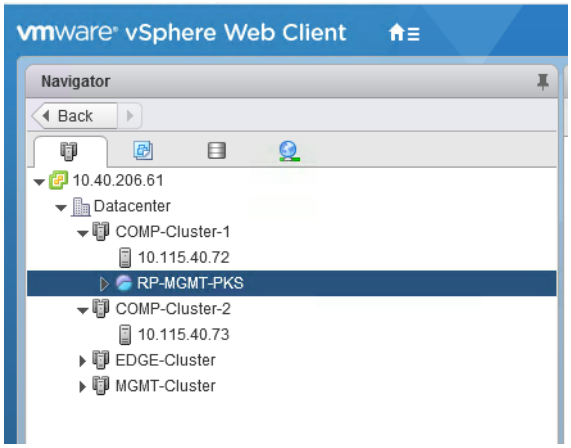


3. Name the resource pool, such as `RP-MGMT-PKS`.



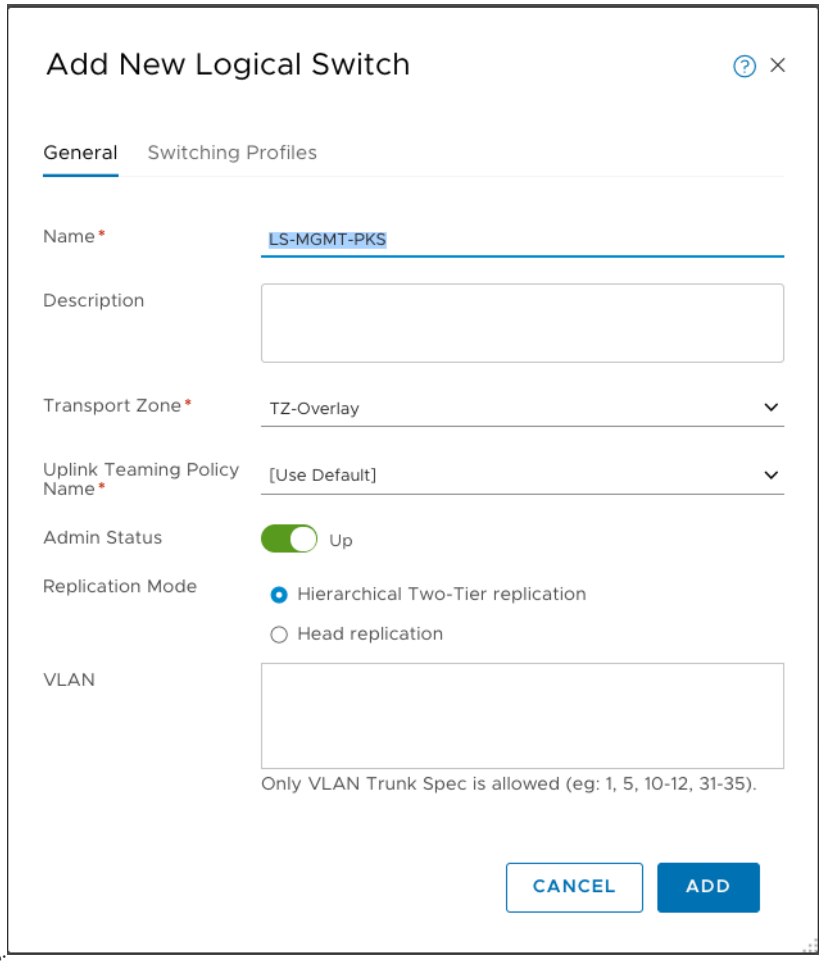
4. Click **OK**

5. Verify resource pool creation.

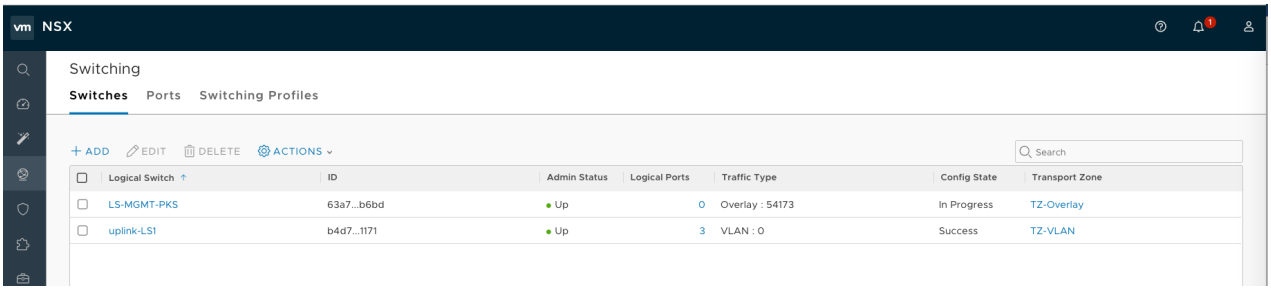


Step 2. Create NSX-T Logical Switch for the PKS Management Plane

1. In NSX Manager, select **Switching > Add**.



2. Create a new logical switch. For example:
3. Click **Add**.
4. Verify logical switch creation.

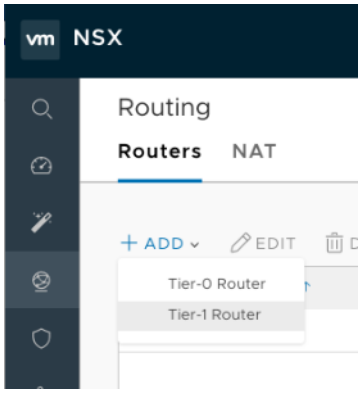


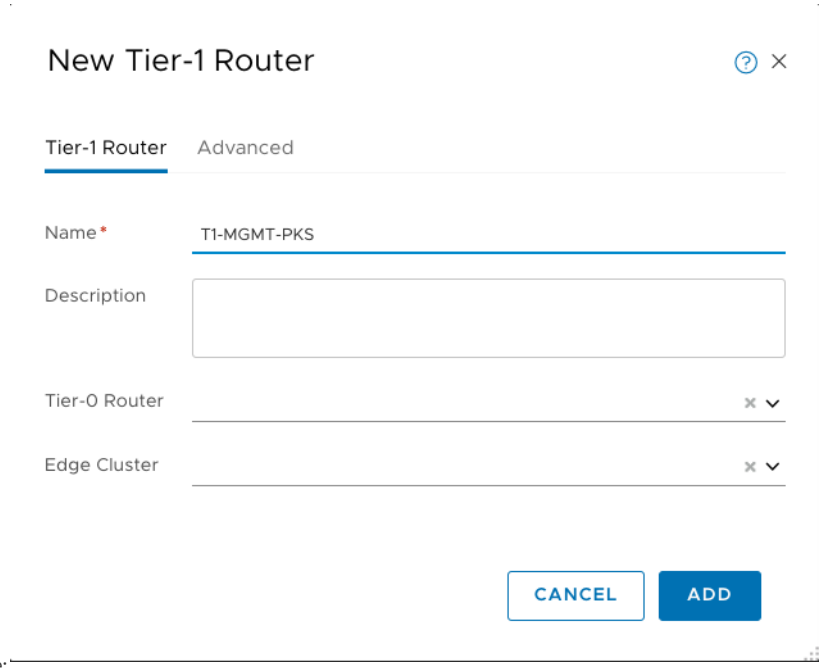
Step 3. Create NSX-T Tier-1 Router for the PKS Management Plane

Defining a T1 router involves creating the router and attaching it to the logical switch, creating a router port, and advertising the routes.

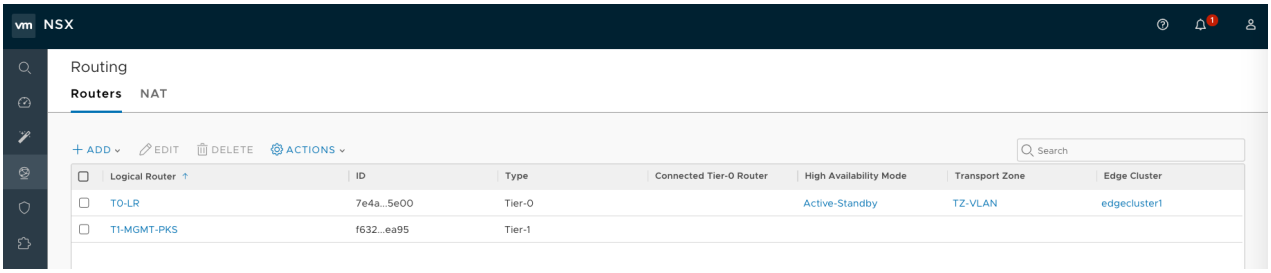
Create T1 Router

1. In NSX Manager, select **Routing > Add > Tier-1 Router**.



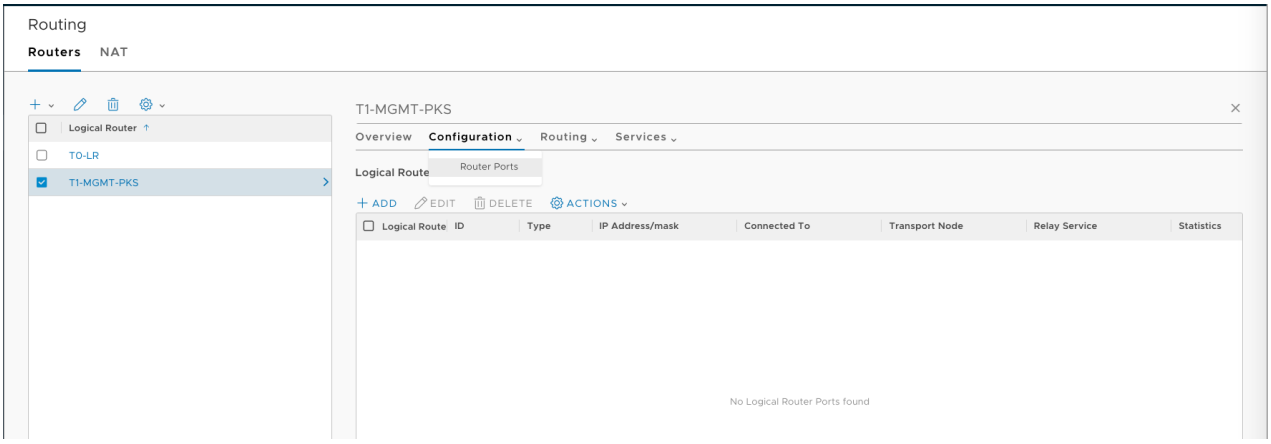


2. Configure the T1 router. For example:
3. Click **Add**.
4. Verify T1 router creation.

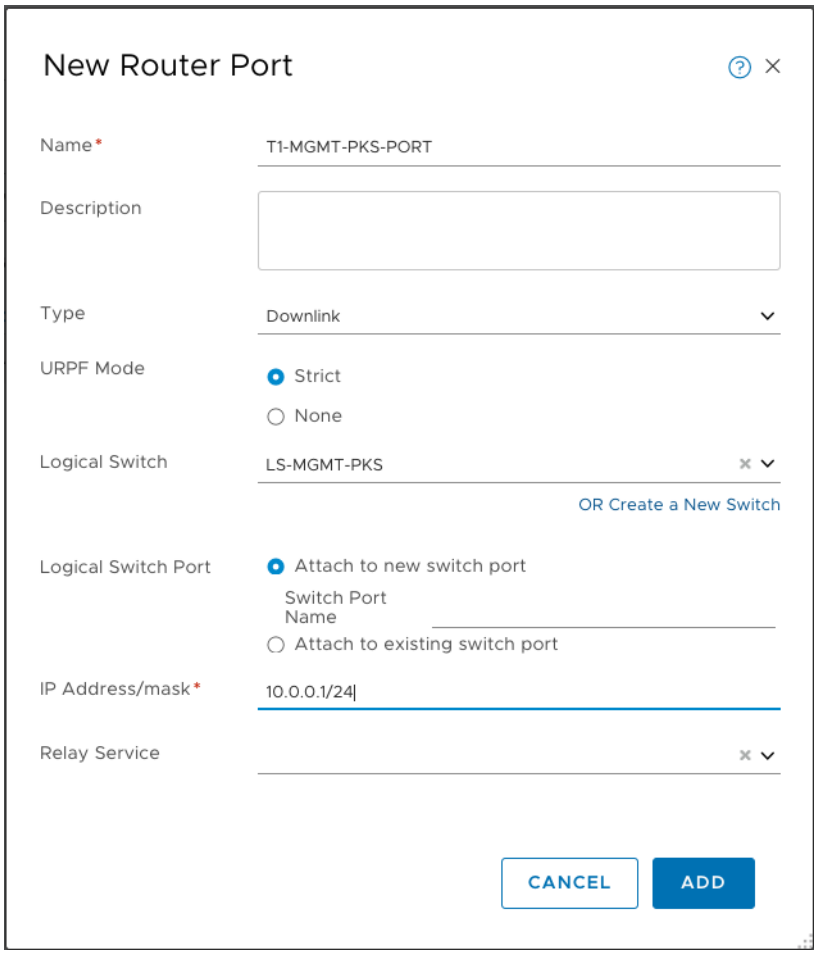


Create T1 Router Port

1. Select the T1 router you created.
2. Select **Configuration > Router Ports**.



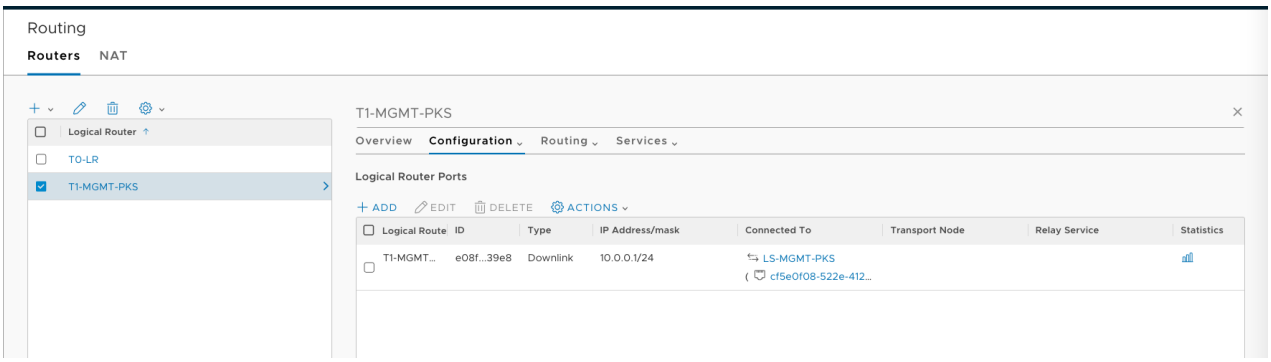
3. Click **Add** and configure the T1 router port. For example:
    - **Name**: `T1-MGMT-PKS-PORT`
    - **Logical Switch**: `select LS-MGMT-PKS`



    - **IP Address/mask**: `10.0.0.1/24`

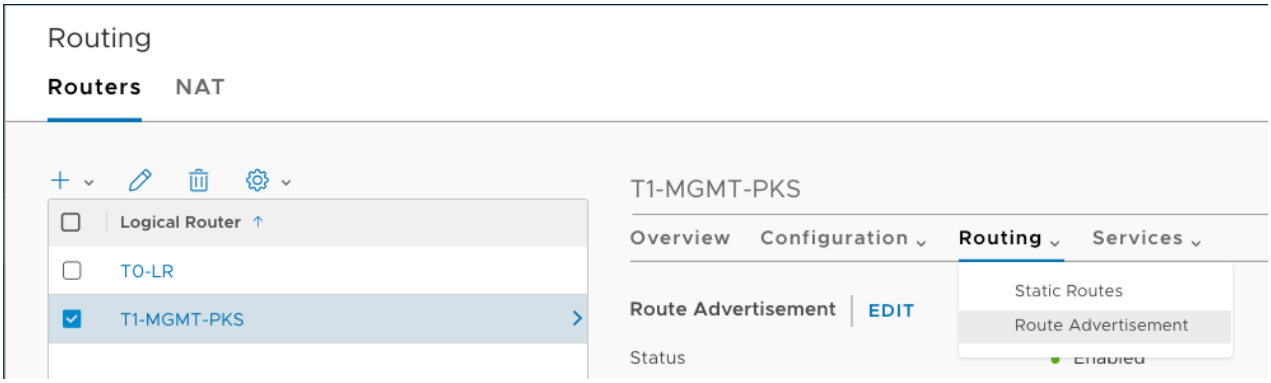
4. Click **Add**.

5. Verify T1 router port creation.

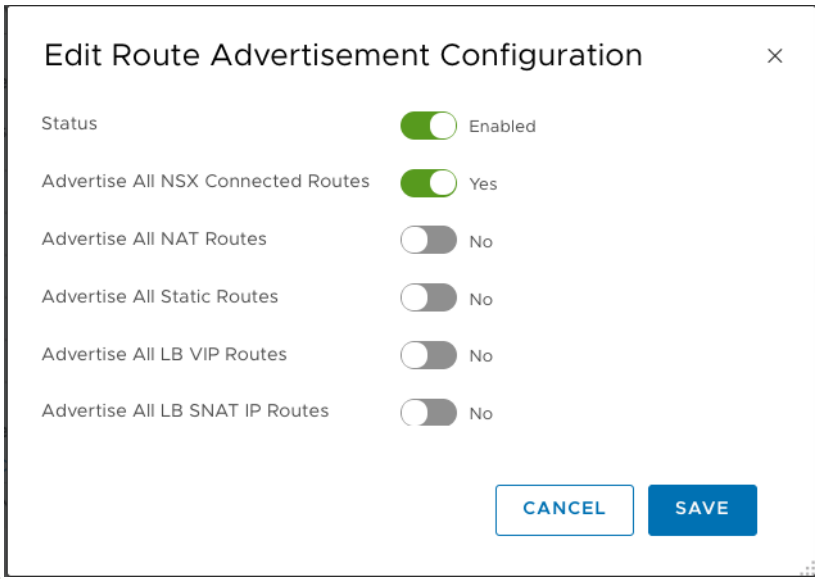


Advertise the T1 Routes

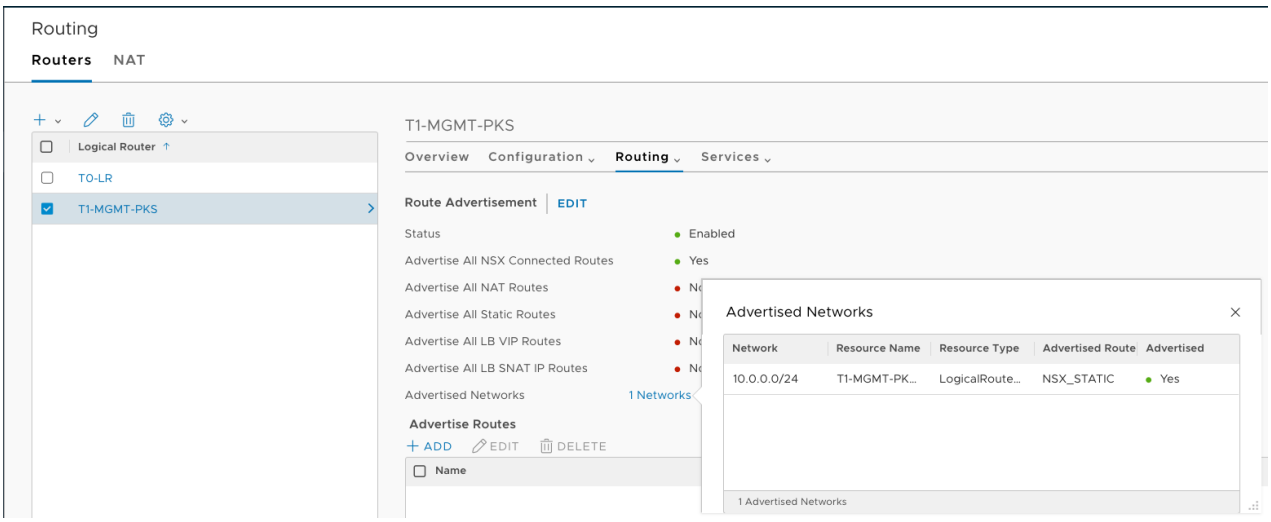
1. Select the T1 router > Routing > Route Advertisement.



2. Advertise the T1 route as follows:
   - **Status**: enabled
   - **Advertise all NSX connected routes**: yes

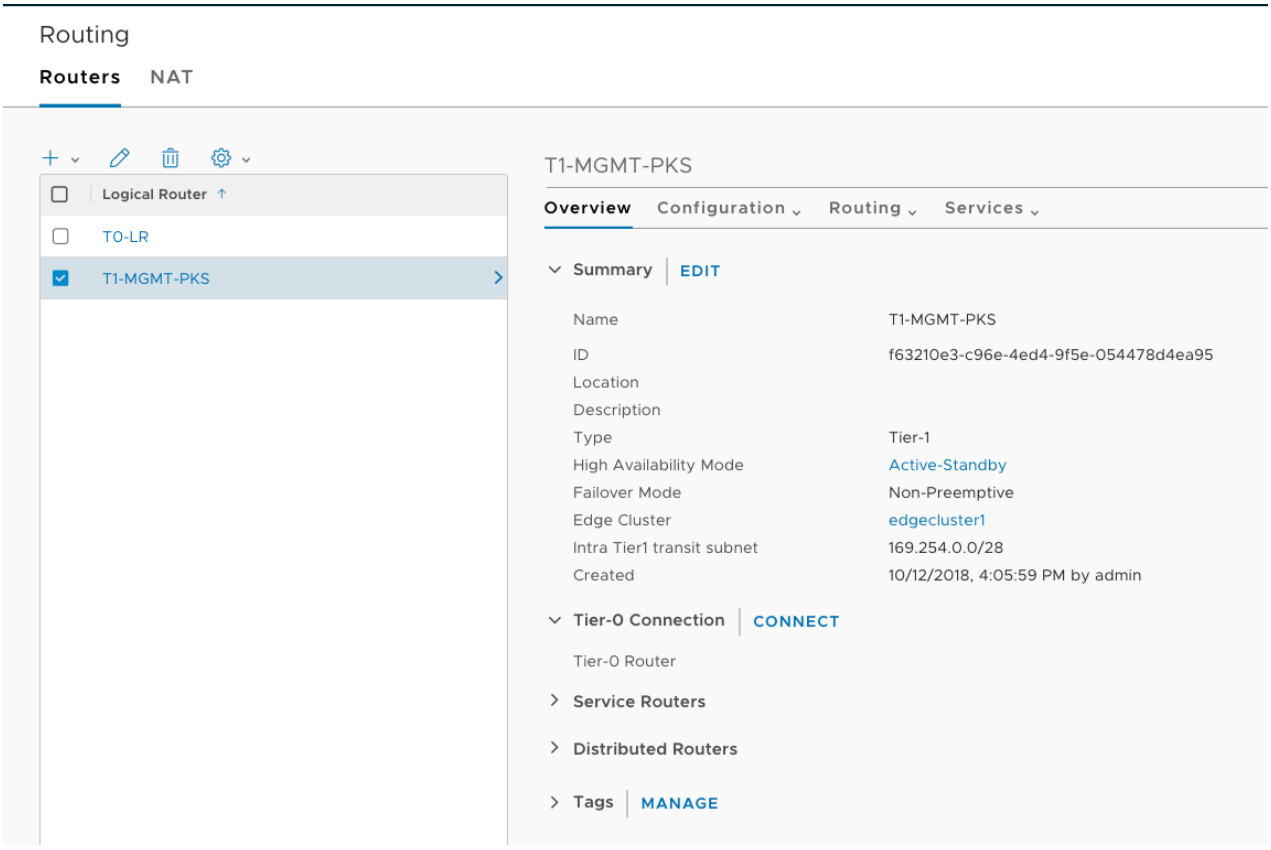


3. Click **Save**.s
4. Verify route advertisement.

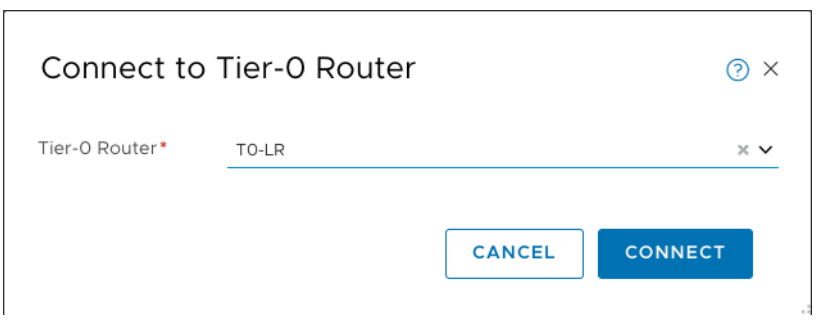


Verify T1 Router

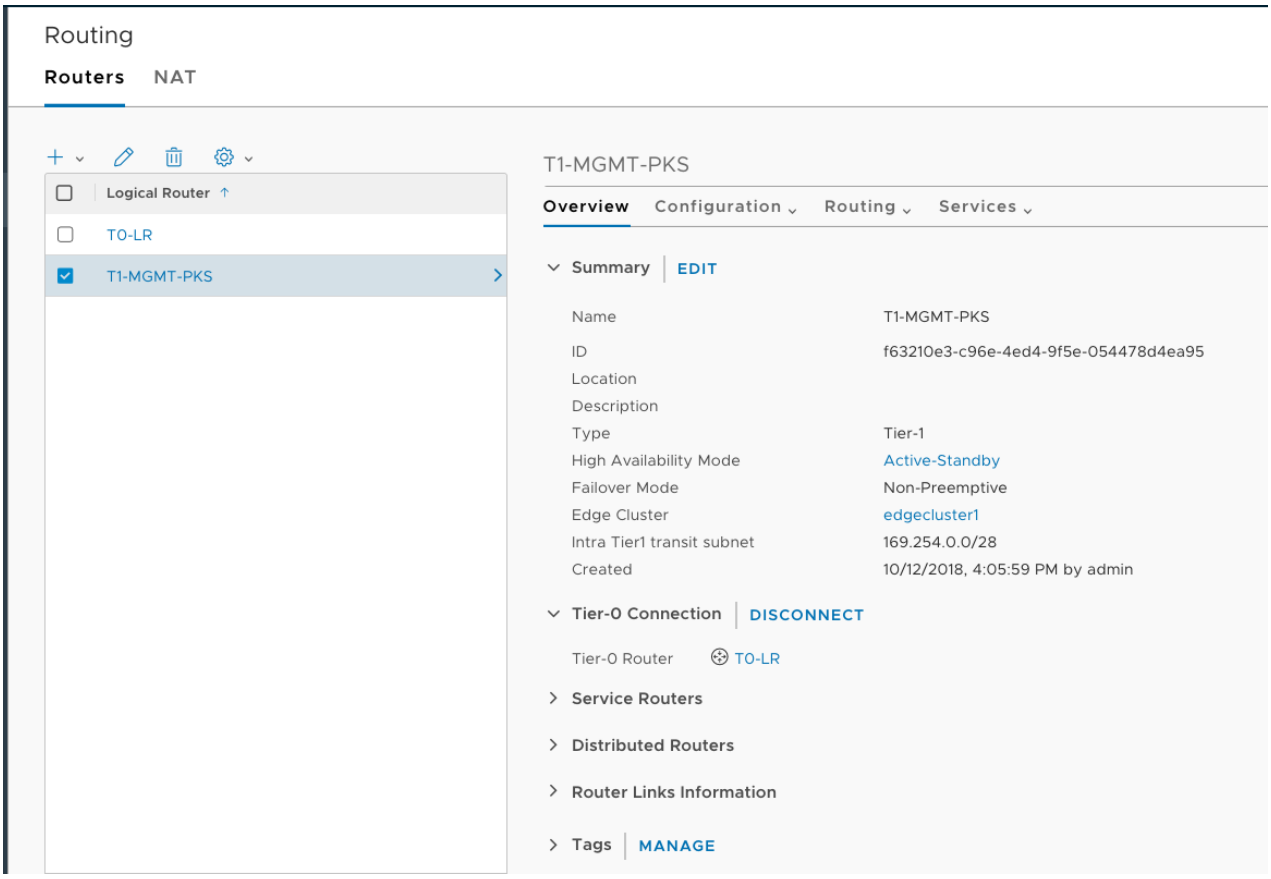
1. Select the **T1 Router > Overview**.



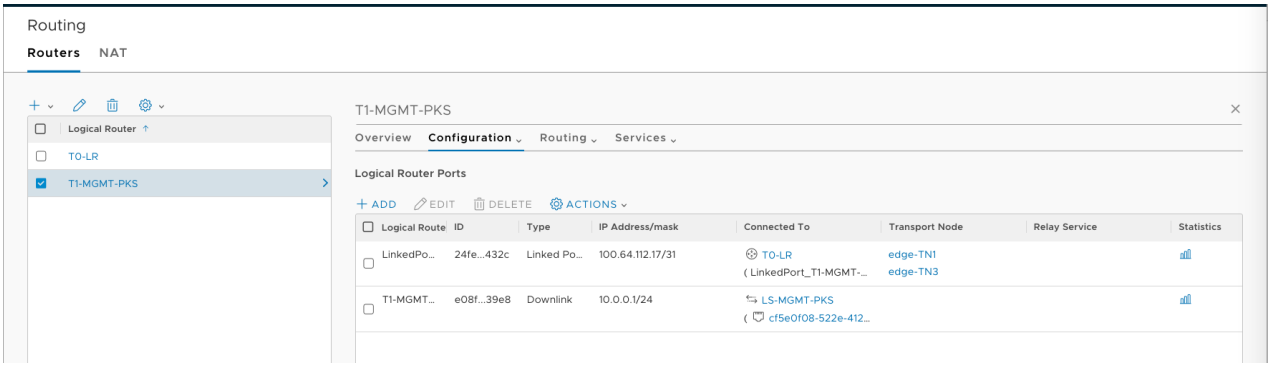
2. Select **Tier-0 Connection > Connect**, then select the T0 router and click **Connect**.



3. Verify connectivity between T1 and T0 routers.



4. Select the **T1 router > Router ports**. The T1 router created for the PKS Management Plane should have 2 ports: one connected to the T0 router, and a second port connected to logical switch defined for the PKS Management Plane. This second port will be the default gateway for all VMs connected to this LS.



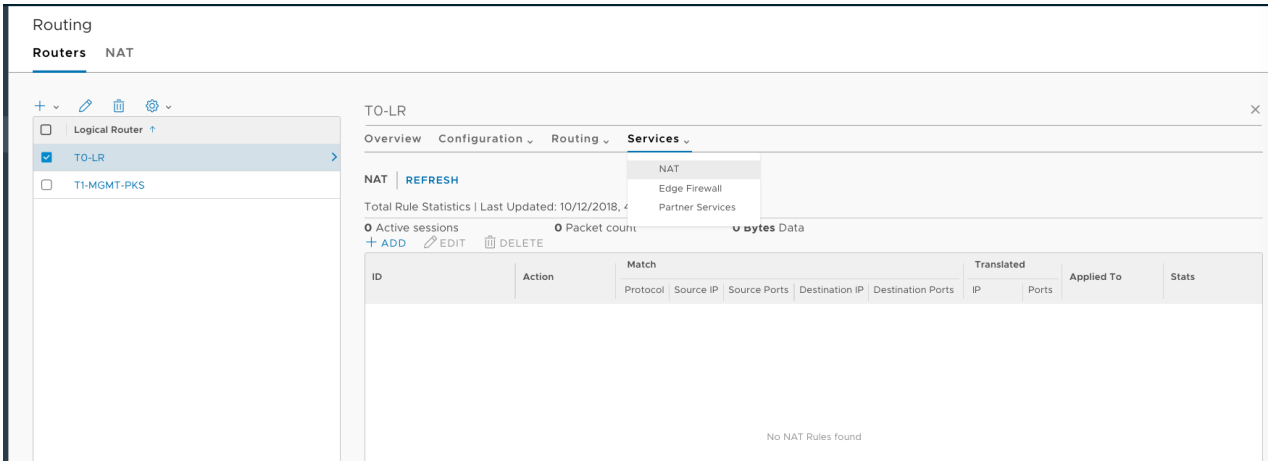
Step 4. Create DNAT Rule on T0 Router for Ops Manager

Create a DNAT rule on the T0 Router to access the Ops Manager Web UI, which is required to deploy PKS.

The Destination NAT (DNAT) rule on the T0 maps an external IP address from the **PKS MANAGEMENT CIDR** to the IP where you deploy Ops Manager on the PKS Management logical switch that you created on the T0 router. For example, a DNAT rule that maps `10.172.1.2` to `172.31.0.2`, where `172.31.0.2` is the IP address you assign to Ops Manager when connected to `ls-pks-mgmt`.

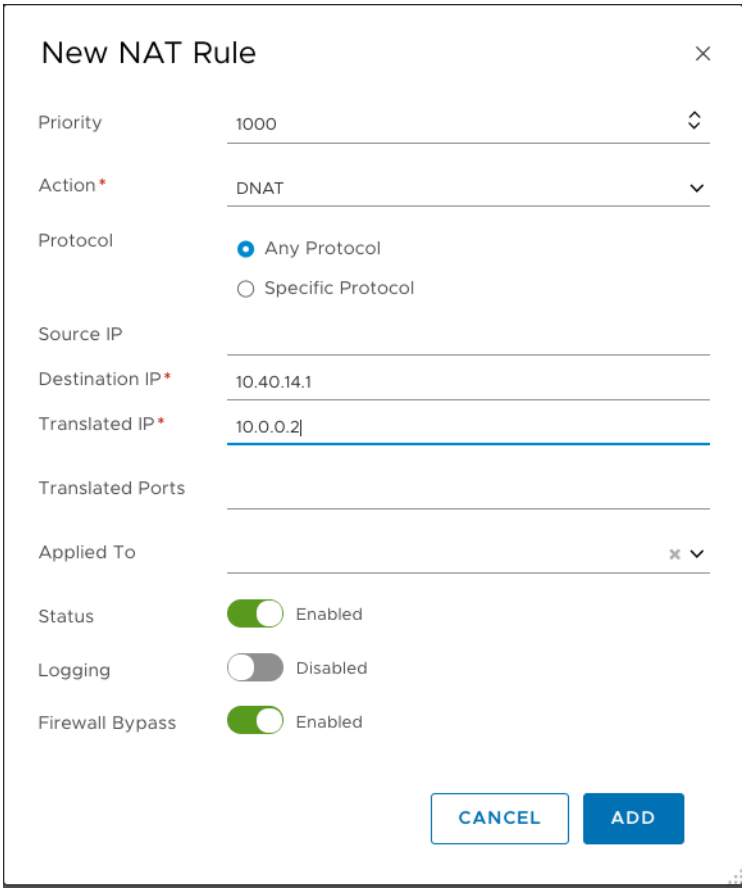
To create a DNAT rule for Ops Manager:

1. In NSX Manager, select **Routing > Routers**.
2. Select the **T0 Router > Services > NAT**.



3. Add and configure a DNAT rule with the routable IP address as the destination and the internal IP address for Ops Manager as the translated IP. For example:

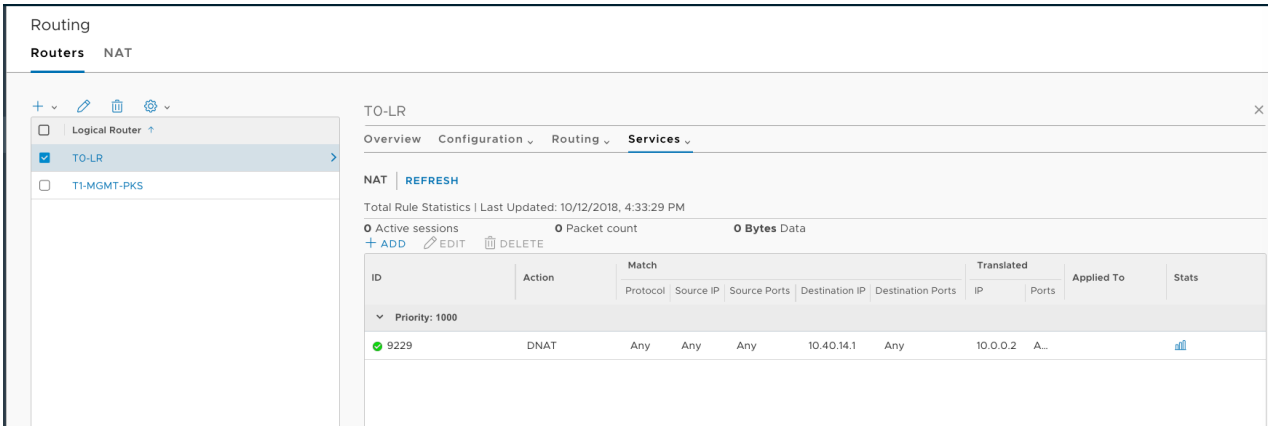
    - **Priority**: 1000
    - **Action**: DNAT
    - **Destination IP**: 10.40.14.1



    - **Translated IP**: 10.0.0.2

4. Click **Add**.

5. Verify the DNAT rule.



Step 5. Create DNAT Rule on T0 Router for Harbor Registry

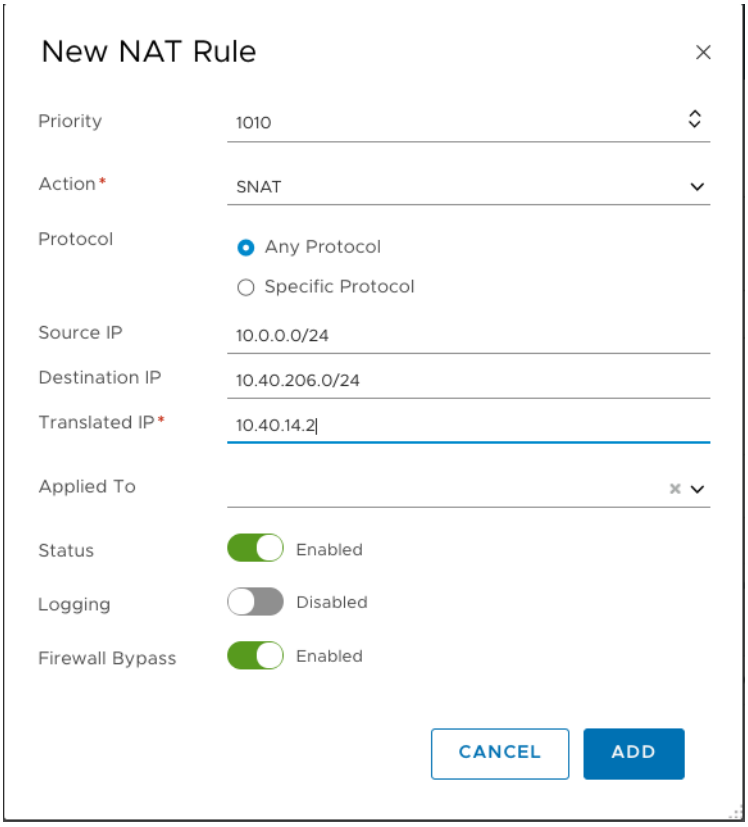
If you are using VMware Harbor Registry with PKS, create a similar DNAT rule on T0 router to access the Harbor Web UI. This DNAT rule maps the private Harbor IP address to a routable IP address from the floating IP pool on the PKS Management network. See Create DNAT Rule in the VMware Harbor Registry documentation for instructions.

Step 6. Create SNAT rule on T0 router for vCenter and NSX Manager

Create a SNAT rule on T0 router for PKS management components to access vCenter and NSX manager. The Source NAT (SNAT) rule on the T0 allows the PKS Management VMs to communicate with your vCenter and NSX Manager environments. For example, a SNAT rule that maps `172.31.0.0/24` to `10.172.1.1`, where `10.172.1.1` is a routable IP address from your **PKS MANAGEMENT CIDR**.

Note: Limit the Destination CIDR for the SNAT rules to the subnets that contain your vCenter and NSX Manager IP addresses.

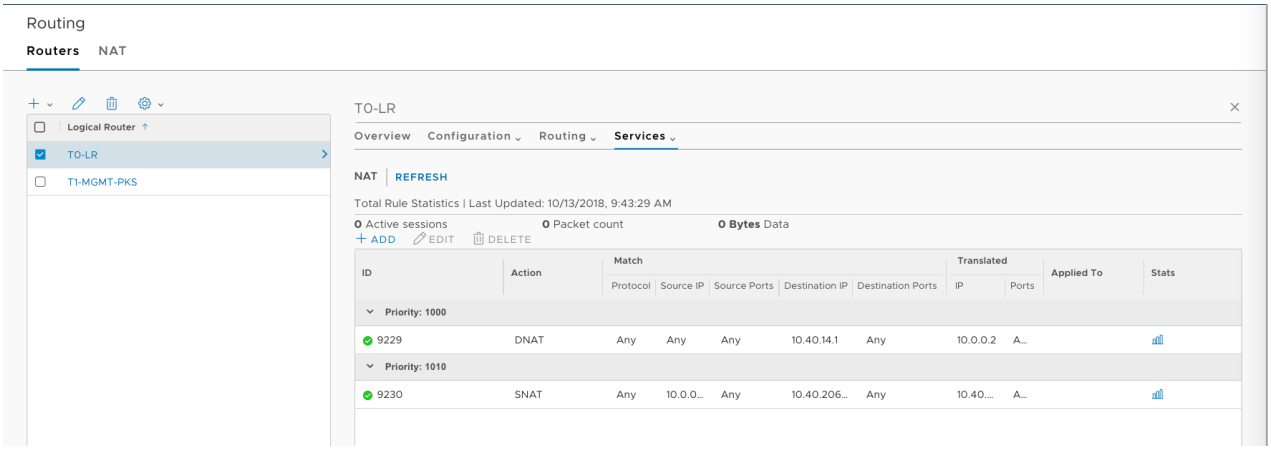
1. Select **T0 router > Services > NAT**.
2. Click ADD and configure the SNAT rule. For example:
    - **Priority**: 1010
    - **Action**: SNAT
    - **Source**: 10.0.0.0/24
    - **Destination IP**: 10.40.206.0/24



- **Translated IP**: 10.40.14.2

3. Click **Add**.

4. Verify SNAT rule creation.

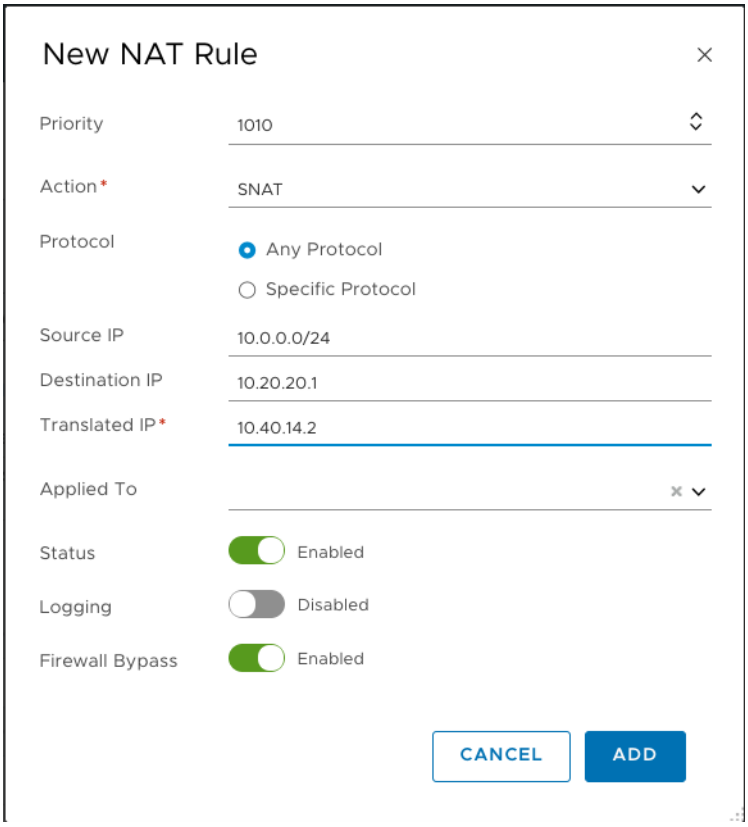


Step 7. Create SNAT Rules on T0 Router for DNS, NTP, and LDAP/AD

1. In NSX Manager, select **T0 router > Services > NAT**.

2. Add a SNAT rule for DNS. For example:

   - **Priority**: 1010
   - **Action**: SNAT
   - **Source**: 10.0.0.0/24
   - **Destination IP**: 10.20.20.1

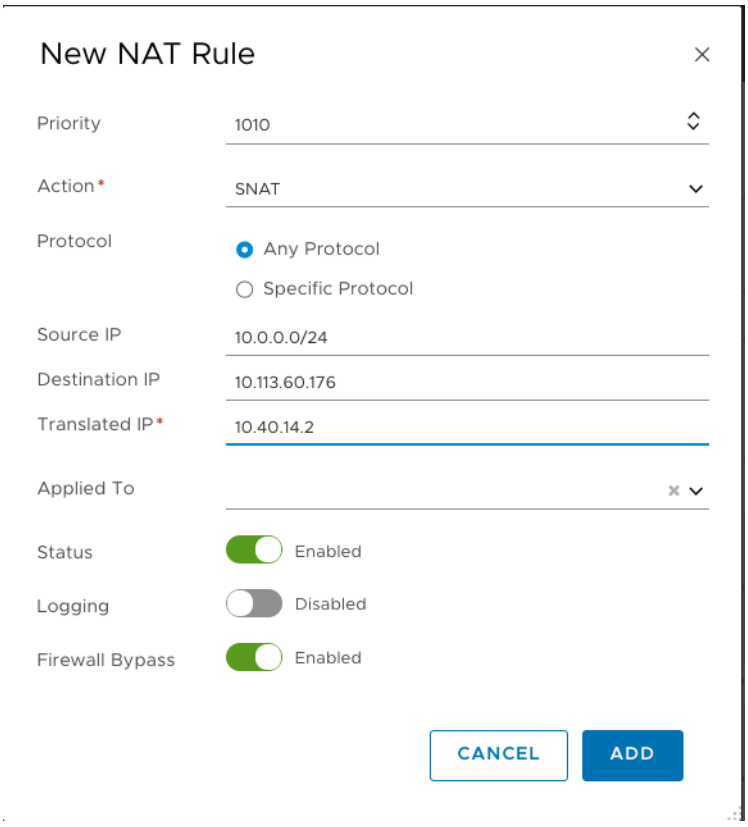


- **Translated IP**: 10.40.14.2

3. Click **Add**.

4. Add a SNAT rule for NTP. For example:

    - **Priority**: 1010
    - **Action**: SNAT
    - **Source**: 10.0.0.0/24
    - **Destination IP**: 10.113.60.176

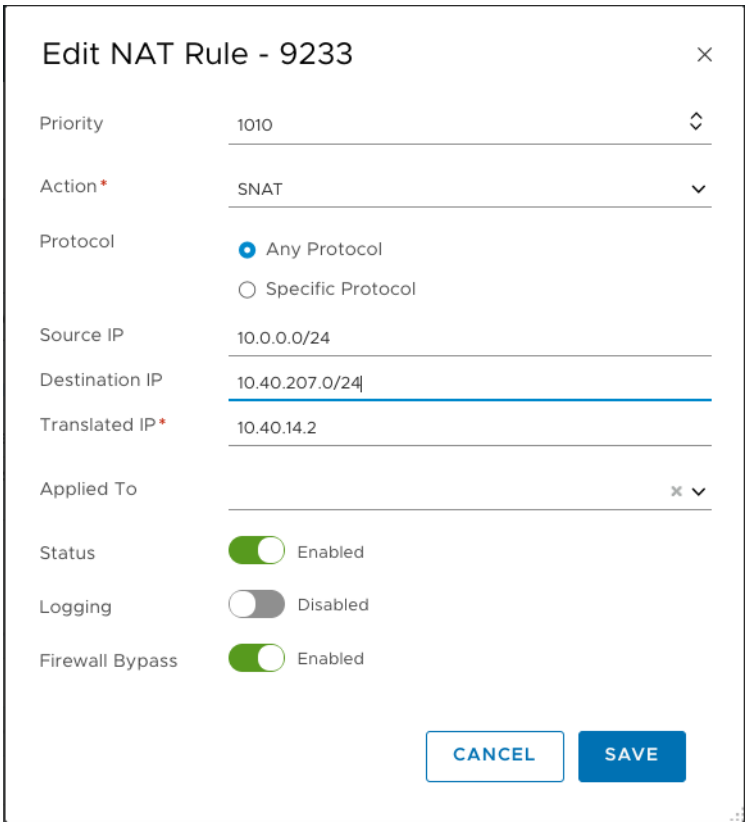


- **Translated IP**: 10.40.14.2

5. Click **Add**.

6. Add a SNAT rule for LDAP/AD. For example:

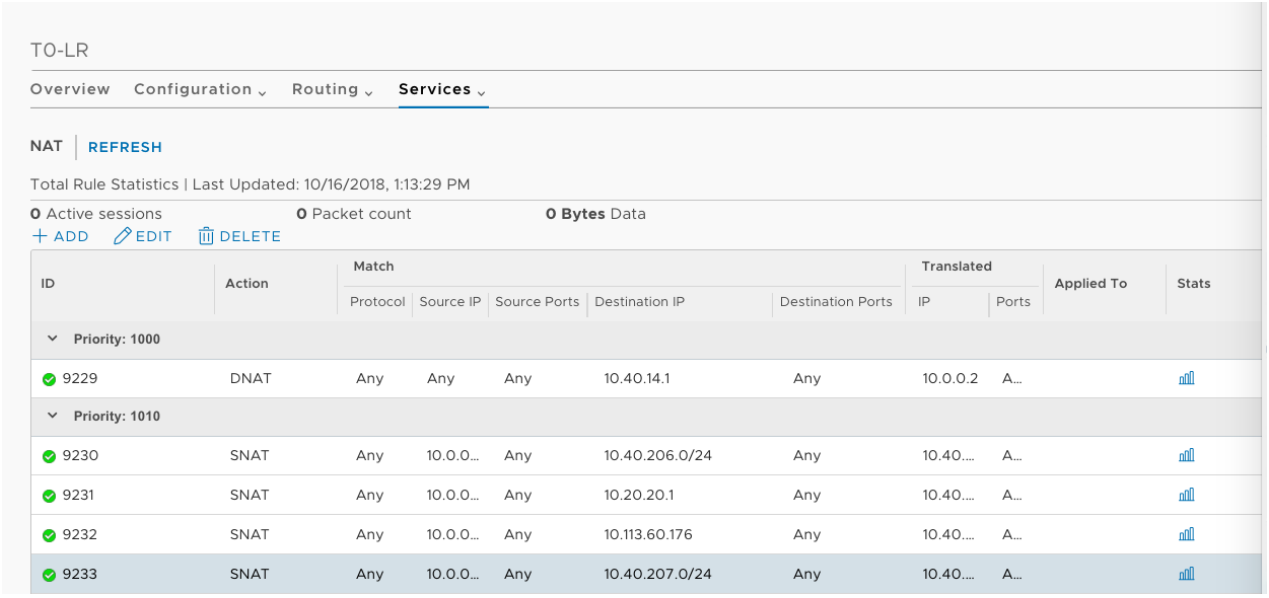
    - **Priority**: 1010
    - **Action**: SNAT
    - **Source**: 10.0.0.0/24
    - **Destination IP**: 10.40.207.0/24



    - **Translated IP**: 10.40.14.2

7. Click **Add**.

8. Verify SNAT rule creation.

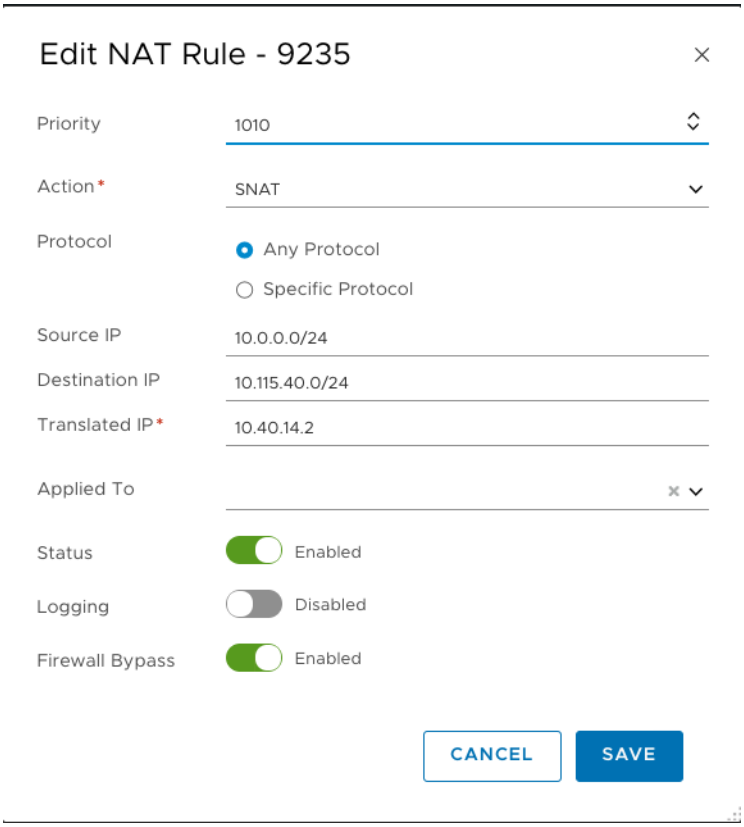


Step 8. Create SNAT Rule on T0 Router for ESXi Hosts

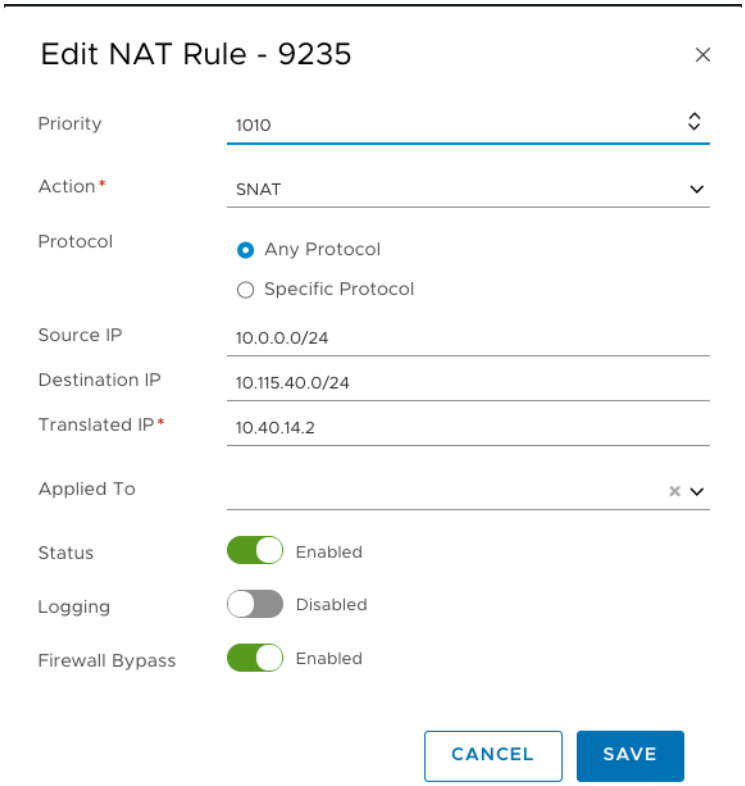
Create a SNAT rule on T0 router for PKS management components to access ESXi Hosts (Management IP). The Destination IP is the Management IP subnet where ESXi Hosts are networked.

Note: Ops Manager and BOSH must use the NFCP protocol to the actual ESX hosts to which it is uploading stemcells. Specifically,**Ops Manager & BOSH Director -> ESXi**.

1. Select **T0 router > Services > NAT**.
2. Click **Add** and configure the SNAT rule. For example:
    - **Priority**: 1010
    - **Action**: SNAT
    - **Destination IP**: 10.115.40.0/24



    - **Translated IP**: 10.40.14.2
3. Click **Add**.



4. Verify SNAT rule creation:

(Optional) Step 9. Create DNAT Rule on T0 Router for External Access to the PKS CLI

This DNAT rule is optional depending on whether or not you need to provide external access to the PKS CLI. If you do need to provide external access, this rule is needed for both NAT and no-NAT modes.

Note: You cannot create this rule until after PKS is installed and the PKS API VM has an IP address.

1. When the PKS installation is completed, retrieve the PKS endpoint by performing the following steps:
    a. From the Ops Manager Installation Dashboard, click the **Pivotal Container Service** tile.
    b. Click the **Status** tab and record the IP address assigned to the `Pivotal Container Service` job.
2. Create a DNAT rule on the shared Tier-0 router to map an external IP from the **PKS MANAGEMENT CIDR** to the PKS endpoint. For example, a DNAT rule that maps `10.172.1.4` to `172.31.0.4`, where `172.31.0.4` is PKS endpoint IP address on the `ls-pks-mgmt` NSX-T Logical Switch.

    Note: Ensure that you have no overlapping NAT rules. If your NAT rules overlap, you cannot reach PKS Management Plane from VMs in the vCenter network.

Next Step

After you complete this procedure, follow the instructions in Creating the PKS Compute Plane.

Creating the PKS Compute Plane

Page last updated:

This section provides instructions for preparing the vSphere and NSX-T infrastructure for the PKS Compute Plane where Kubernetes clusters run.

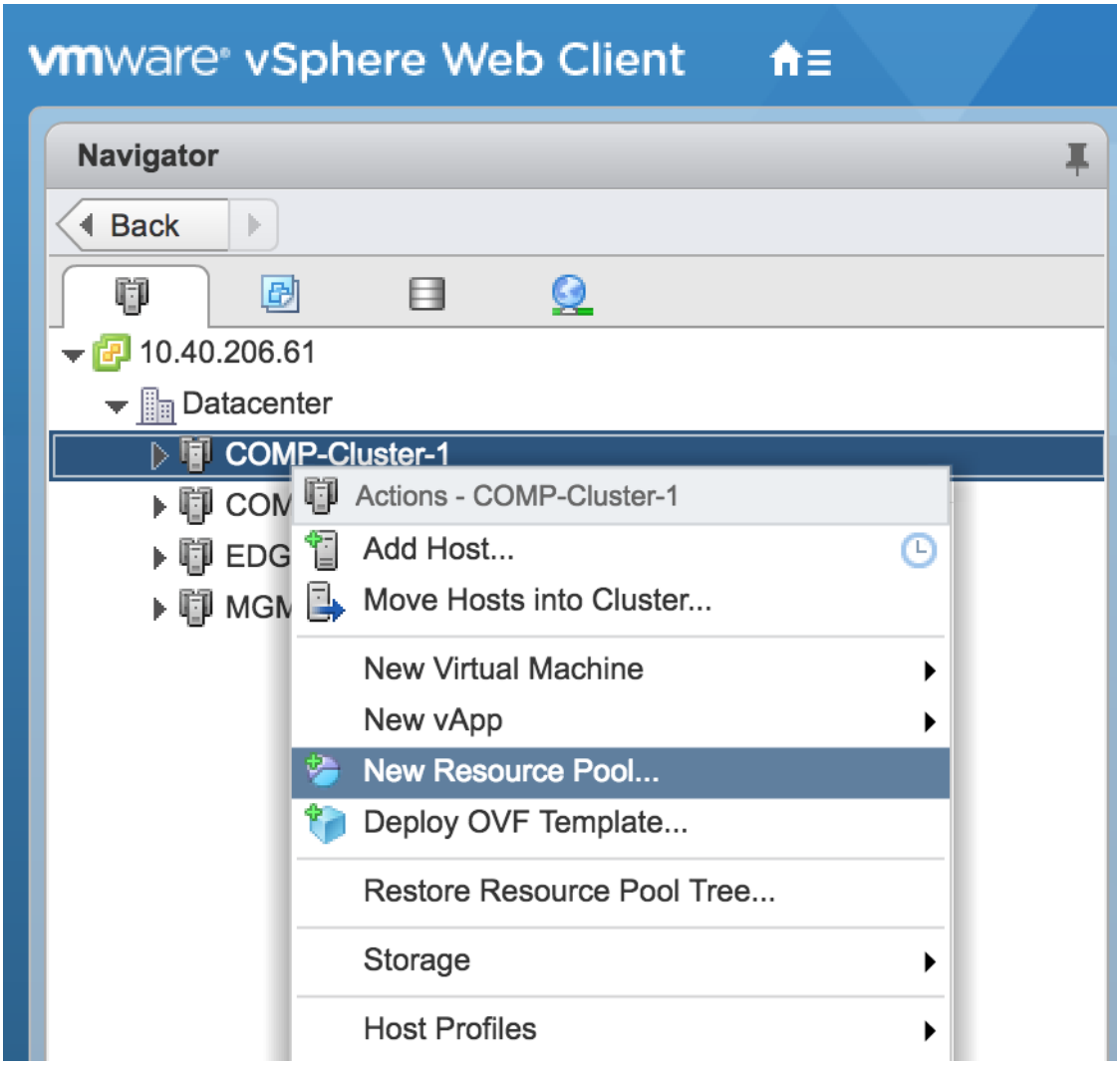
Prerequisites

Before you begin this procedure, ensure that you have successfully completed all preceding steps for installing PKS on vSphere with NSX-T, including:

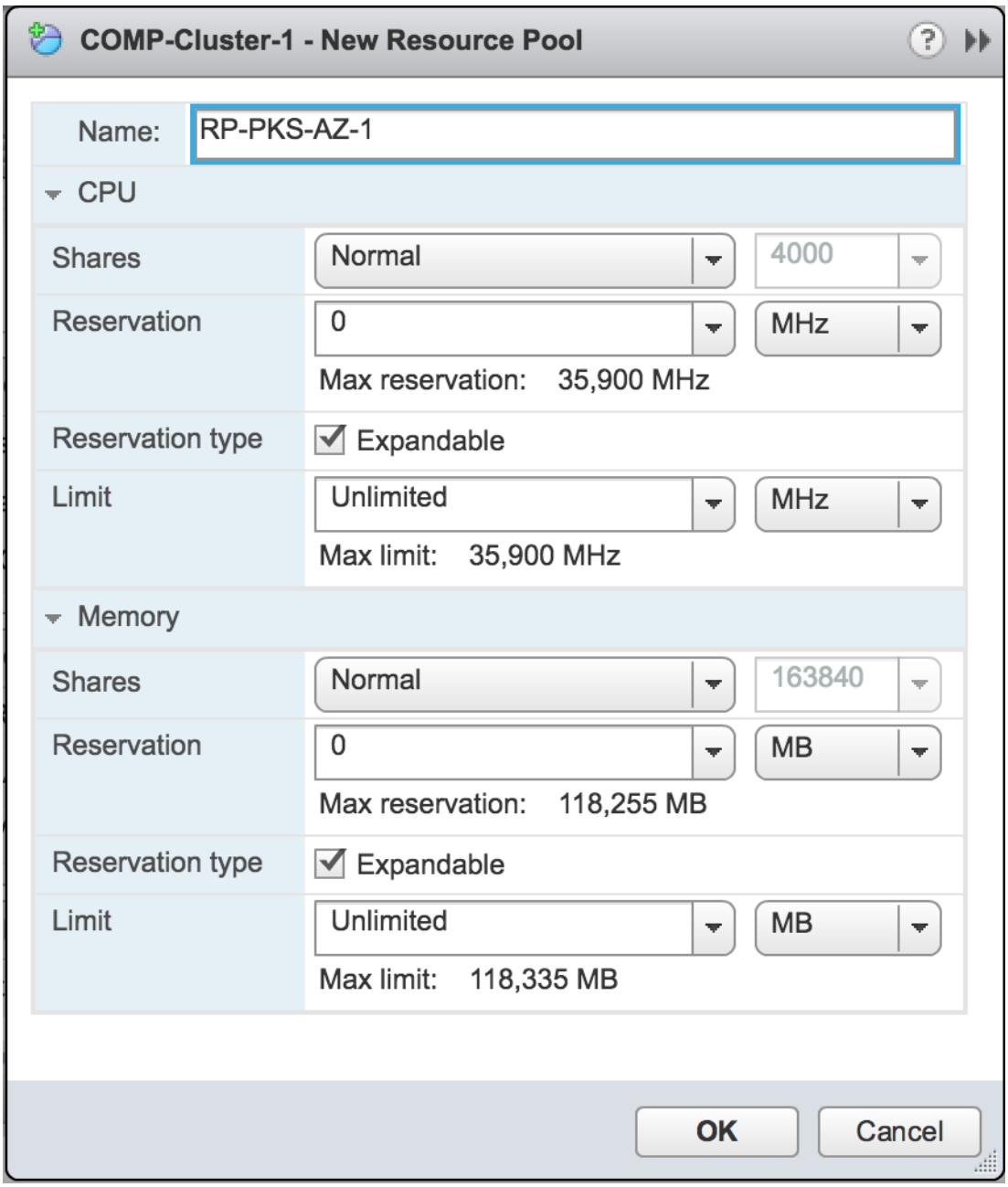
- Deploying NSX-T for PKS
- Creating the PKS Management Plane

Step 1: Create vSphere Resource Pools for AZ-1 and AZ-2

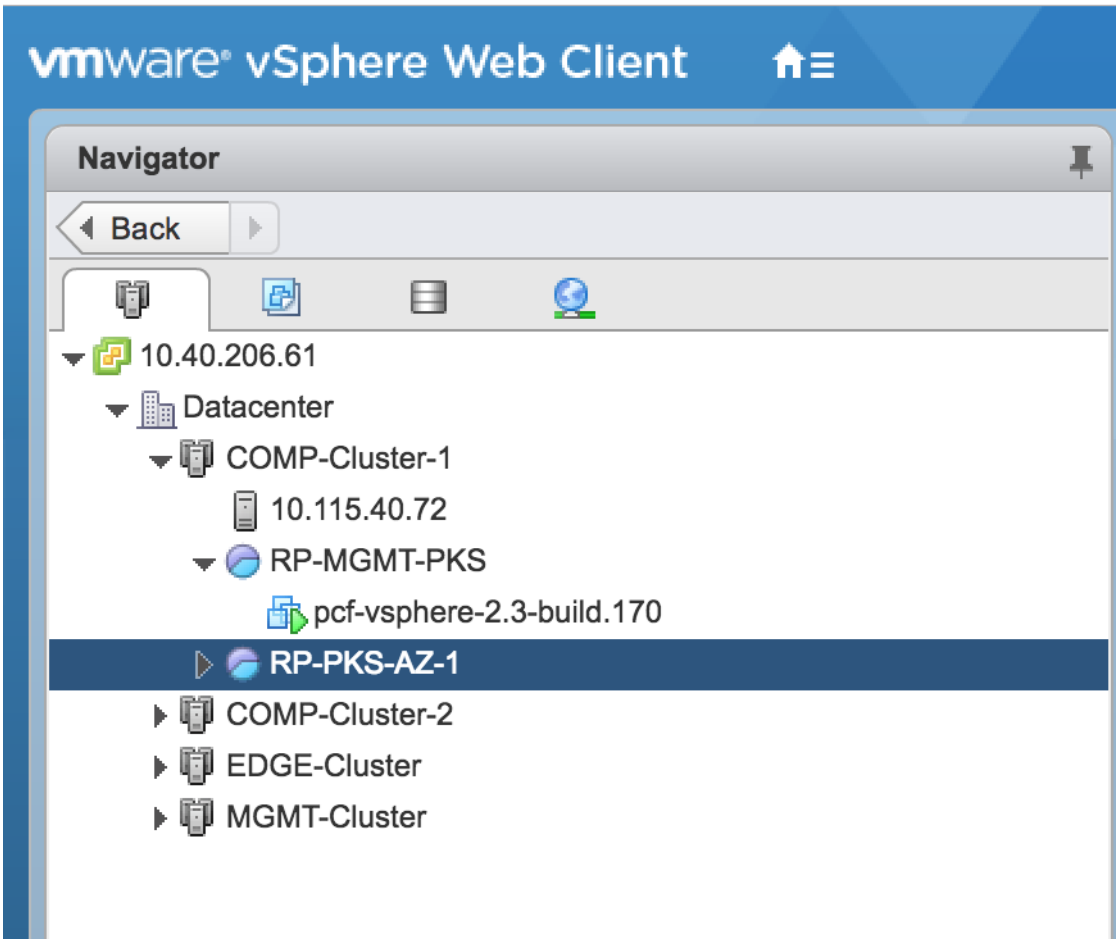
1. Log in to vCenter for your vSphere environment.
2. Select **Compute Cluster > New Resource Pool**.



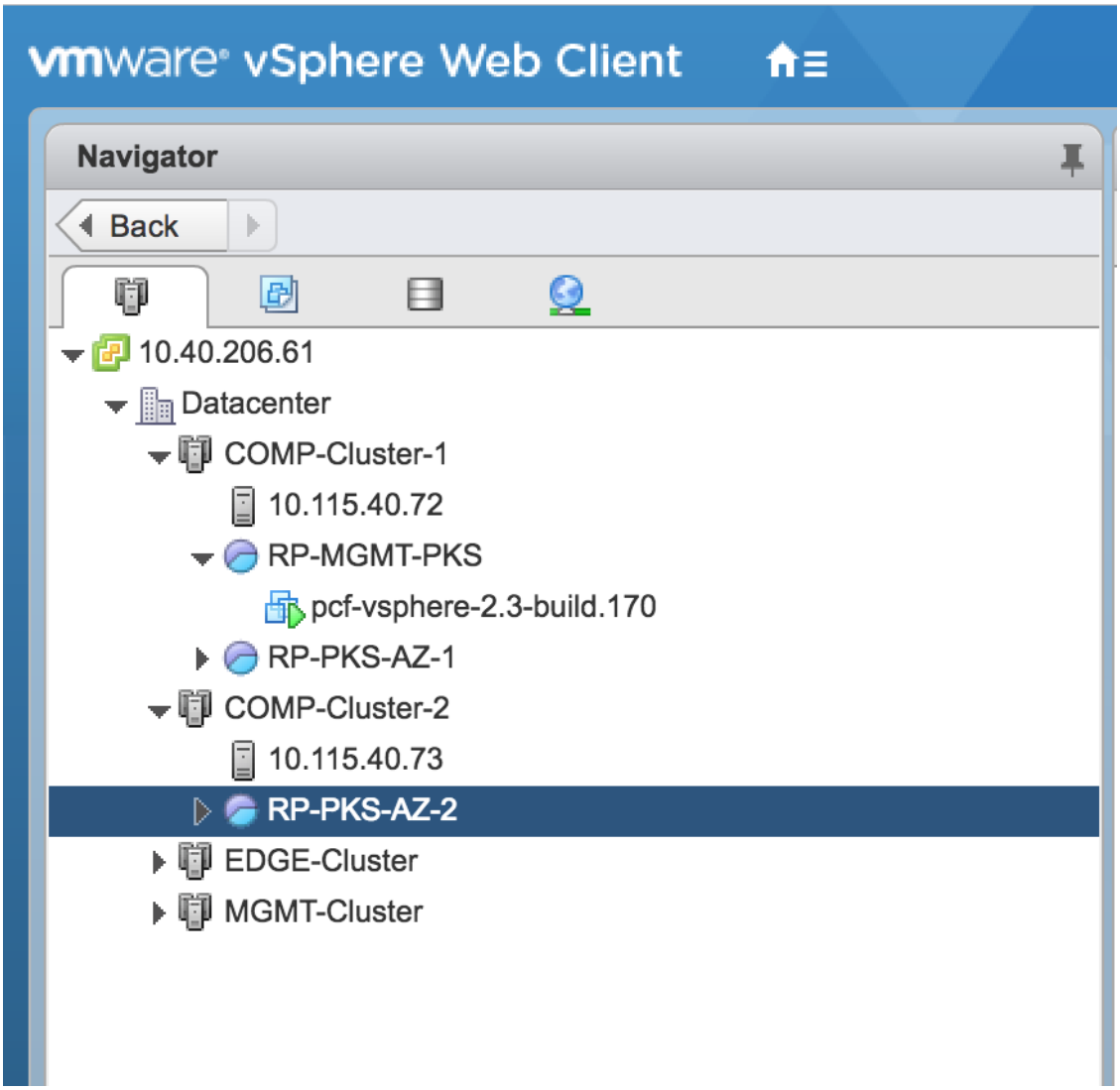
3. Name the resource pool, such as `RP-PKS-AZ-1`.



4. Click **OK** and verify resource pool creation:



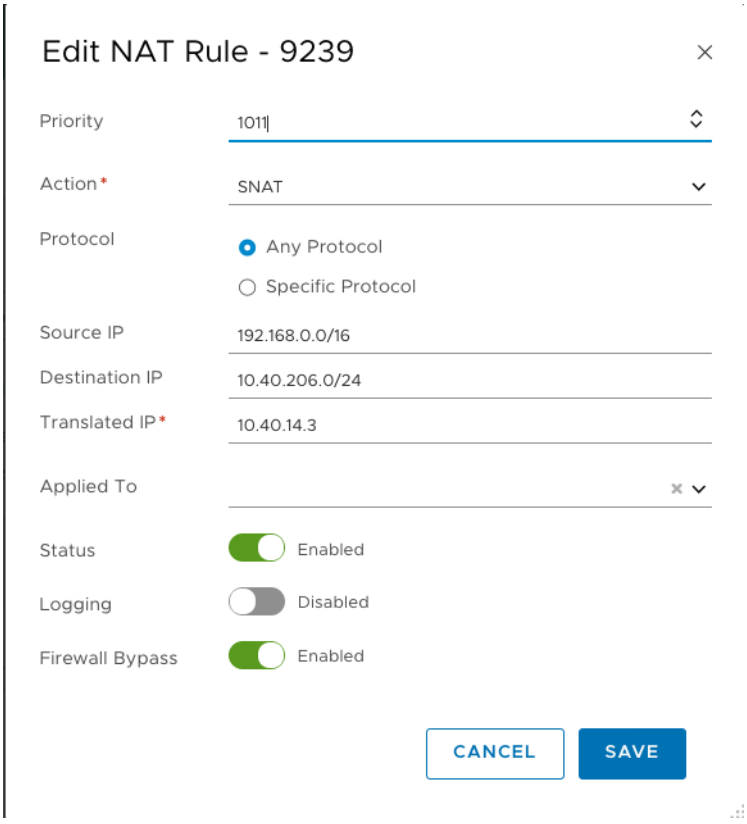
5. Repeat the same operation for Compute Cluster 2 (`RP-PKS-AZ-2`):



Step 2: Create SNAT rule on T0 Router for Kubernetes Access to NSX Manager

Create a SNAT rule on T0 router for K8s Master Nodes (hosting NCP) to reach NSX Manager.

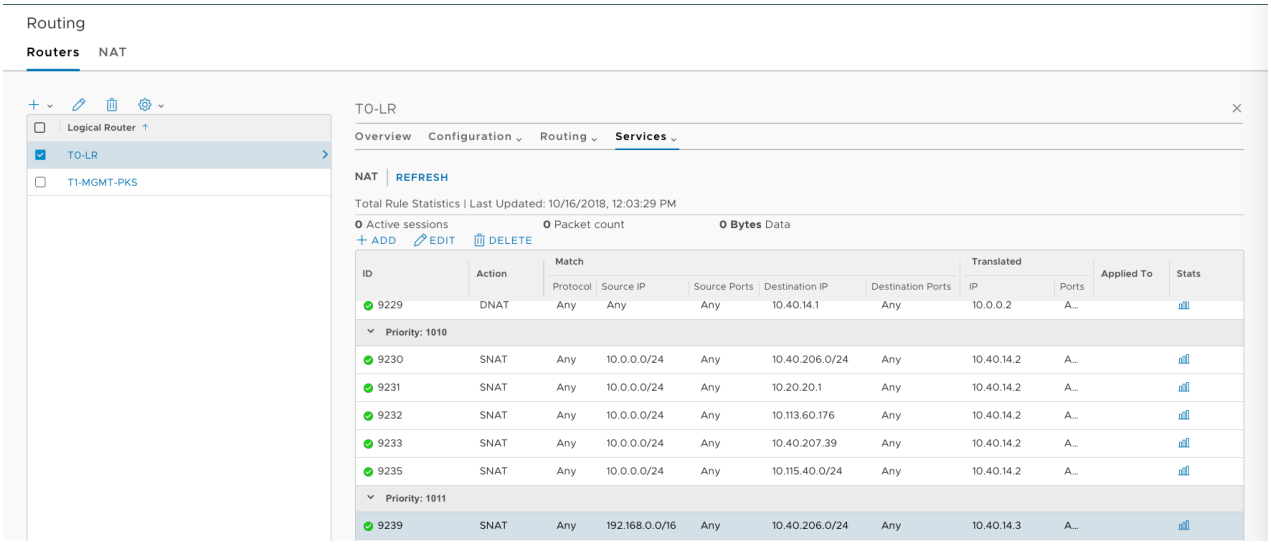
1. Select the **T0 router > Services > NAT**.
2. Click **ADD** and configure the SNAT rule. For example:
    - **Priority**: 1011
    - **Action**: SNAT
    - **Source**: 192.168.0.0/16
    - **Destination IP**: 10.40.206.0/24



- **Translated IP**: 10.40.14.3

3. Click **Save**.

4. Verify SNAT rule creation:



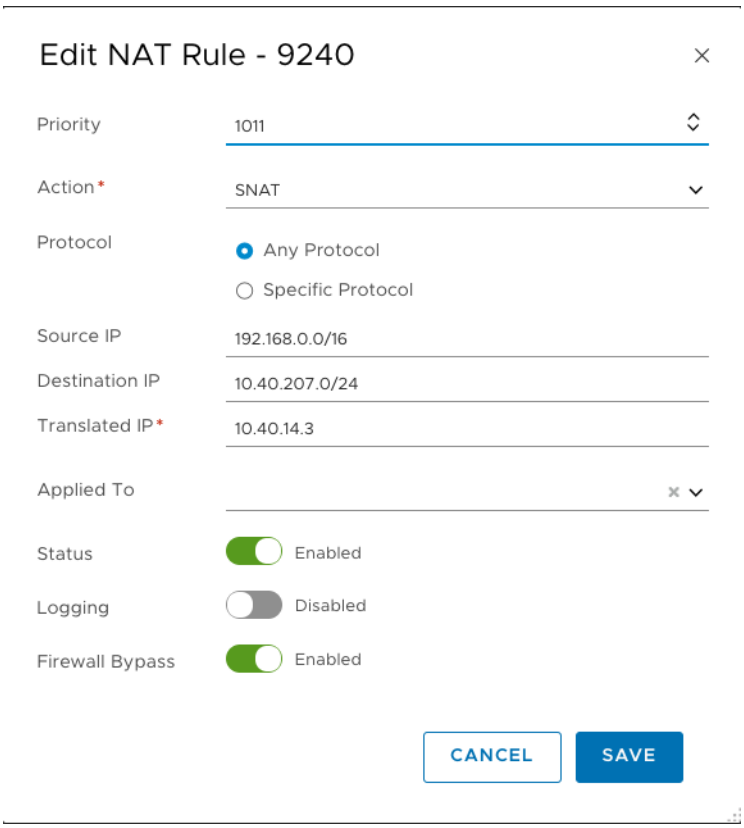
Step 3: Create SNAT Rule on T0 Router for Kubernetes Access to LDAP/AD

Create a SNAT rule on T0 router for K8s Master Nodes (hosting NCP) to reach AD (LDAP) Server (if necessary).

1. In NSX Manager, select the **T0 router > Services > NAT**.

2. Add an SNAT rule for K8S Master Node access to LDAP/AD. For example:

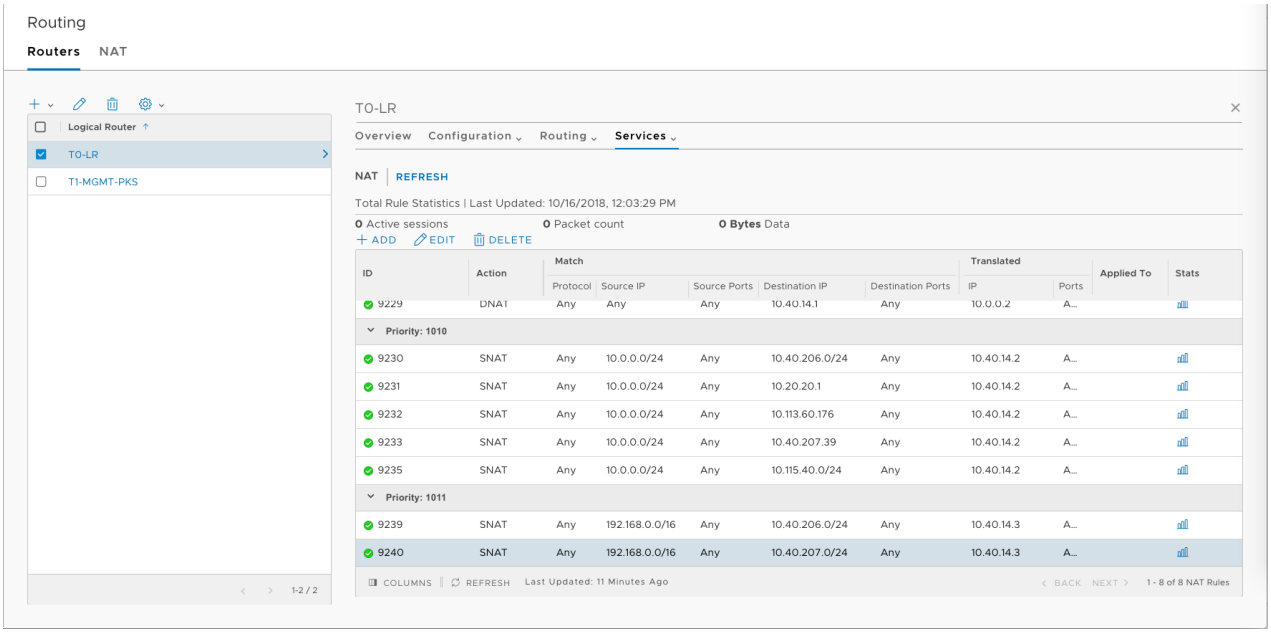
   - **Priority**: 1011
   - **Action**: SNAT
   - **Source**: 192.168.0.0/16
   - **Destination IP**: 10.40.207.0/24



- **Translated IP:** 10.40.14.3

3. Click **Save**.

4. Add and verify SNAT rule creation:



Next Step

After you complete this procedure, follow the instructions in Deploying Ops Manager with NSX-T for PKS.

Deploying Ops Manager with NSX-T for PKS

Page last updated:

This topic provides instructions for deploying Ops Manager on VMware vSphere with NSX-T integration for use with PKS.

Note: For security purposes, VMware requires a dedicated instance of Ops Manager for use with PKS. Do not deploy Pivotal Application Service (PAS) on the same instance of Ops Manager as PKS. For more information, see PAS and PKS Deployments with Ops Manager.

Prerequisites

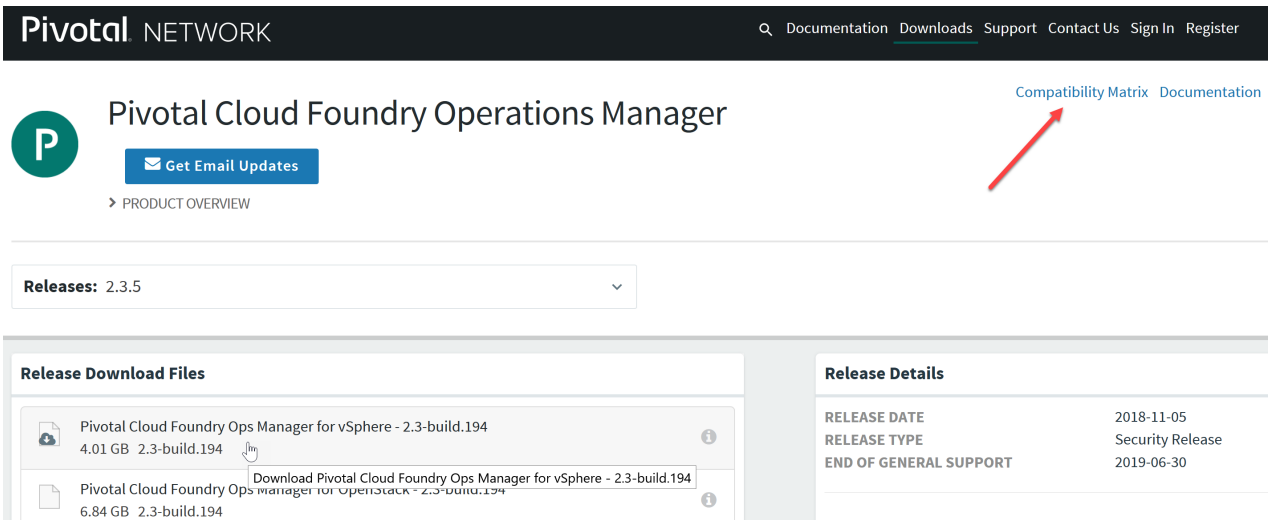
- Deploy NSX-T for PKS
- Create PKS Management Plane
- Create PKS Compute Plane

Deploy Ops Manager for PKS

1. Before starting, refer to the PKS Release Notes for supported Ops Manager versions for PKS. Or, download the Compatibility Matrix from the Ops Manager download page.

2. Before starting, refer to the known issues in the PCF Ops Manager Release v2.3 Release Notes or the PCF Ops Manager Release v2.4 Release Notes.

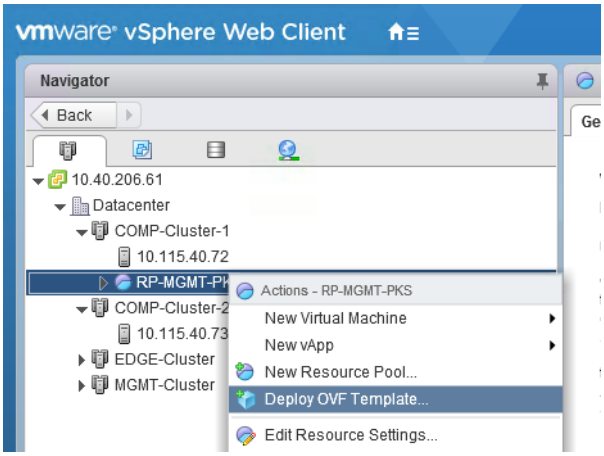
3. Download the Pivotal Cloud Foundry Ops Manager for vSphere `.ova` file at Pivotal Network. Use the dropdown menu to select the supported Ops Manager release.
   Ops Manager for vSphere is provided as an OVA file (`pcf-vsphere-2.3-build.170.ova` , for example) that you import into your vSphere environment. An OVA file is a template for a VM.



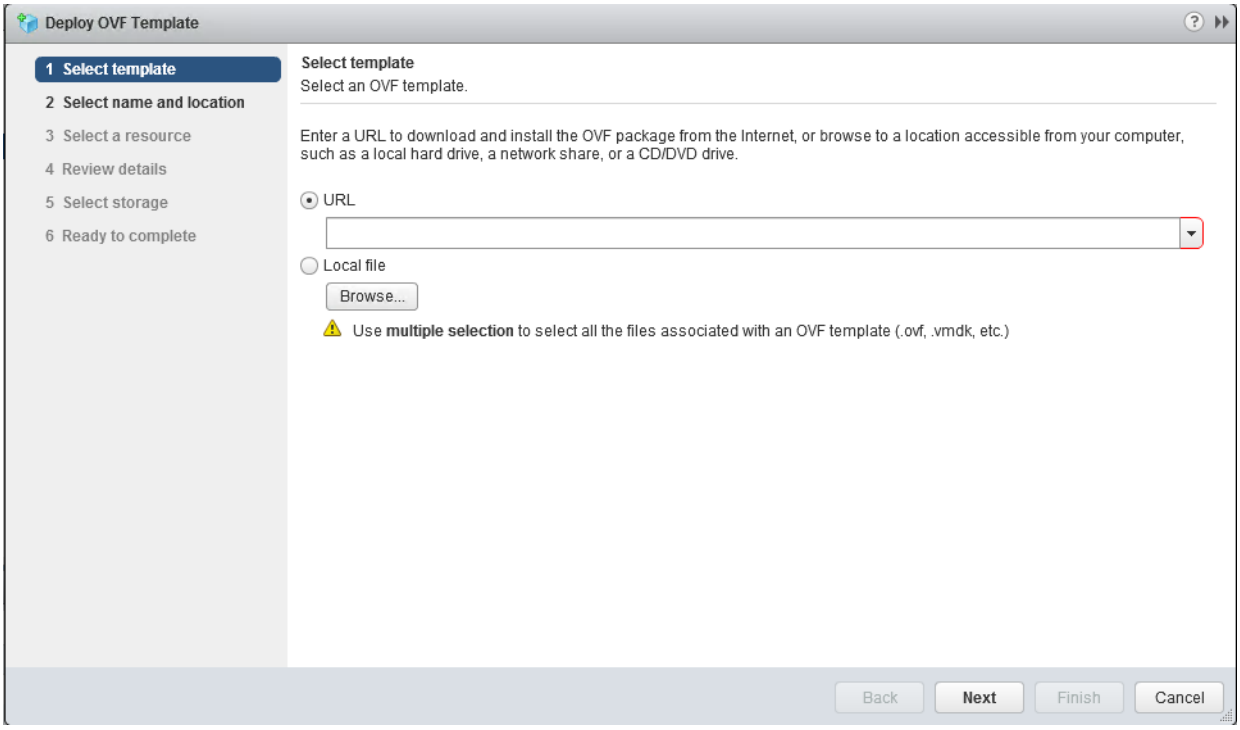
4. Log into vCenter using the vSphere Web Client (FLEX) to deploy the Ops Manager OVA. This can also be done using the using the vSphere Client (HTML5), the OVFTool, or the PowerCLI.

5. Select the Resource Pool defined for the PKS Management Plane. See Create PKS Management Plane if you have not defined the PKS Management Resource Pool.

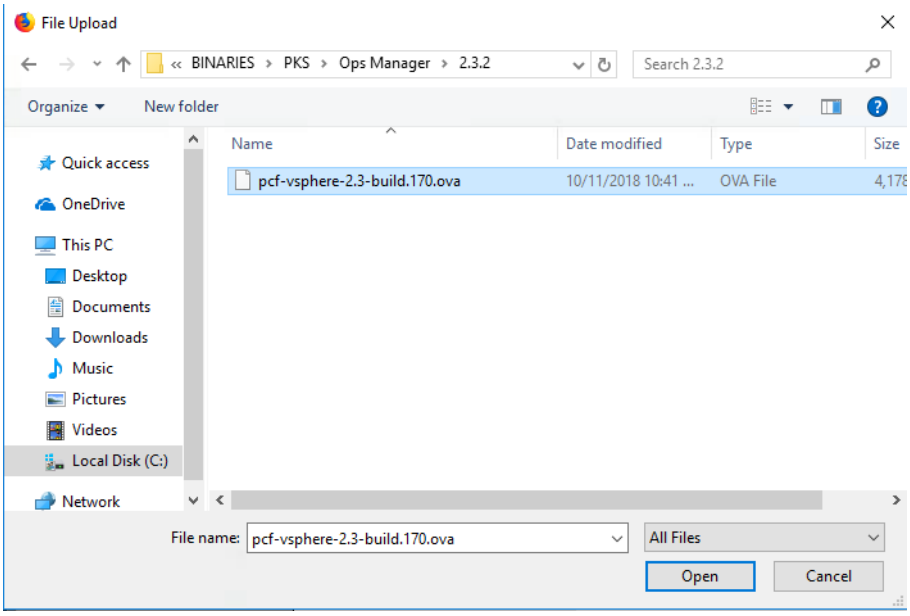
6. Right click the PKS Management Plane Resource Pool and select **Deploy OVF Template.**



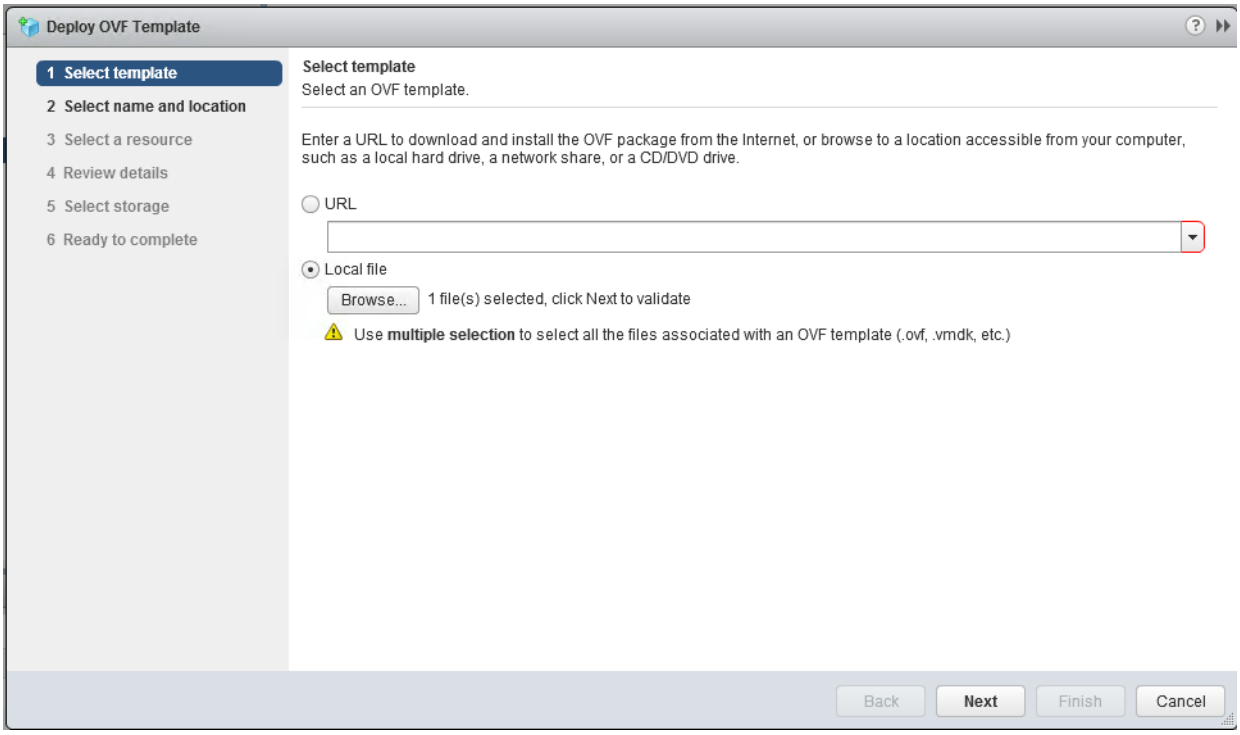
7. At the **Select template** screen, click **Browse**.



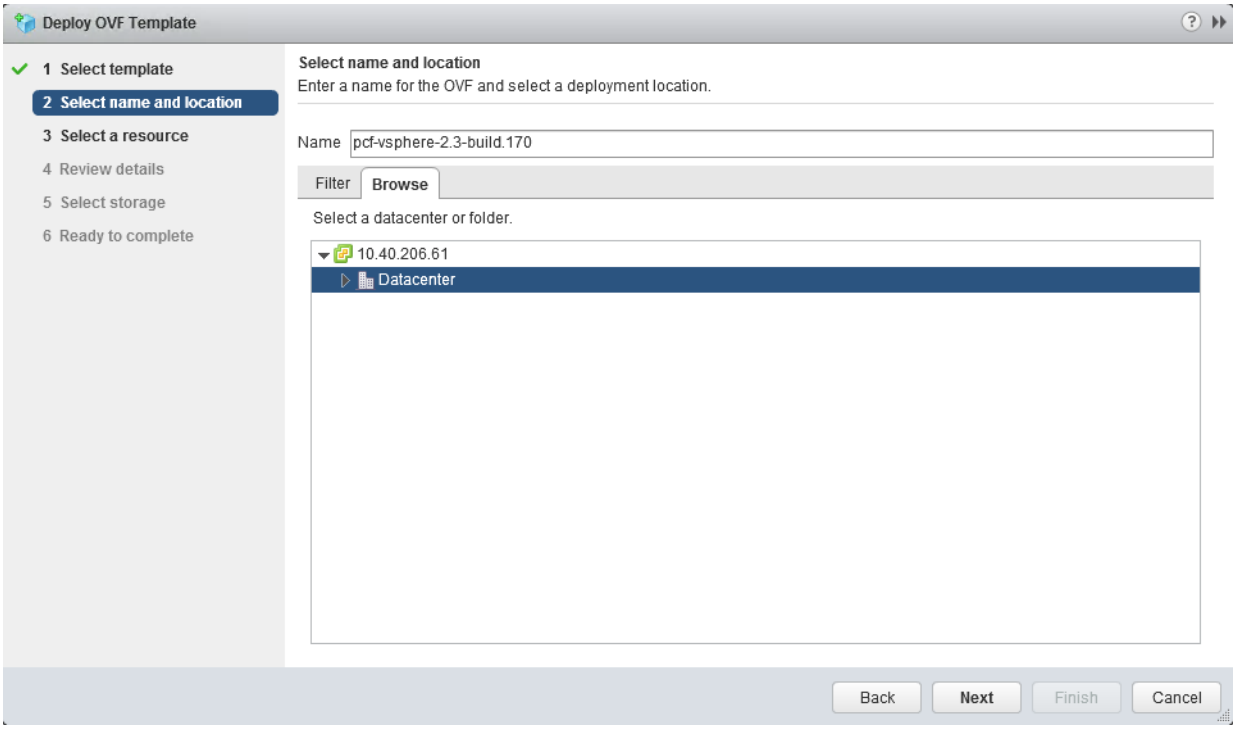
8. Select the Ops Manager OVA file you downloaded and click **Open**.



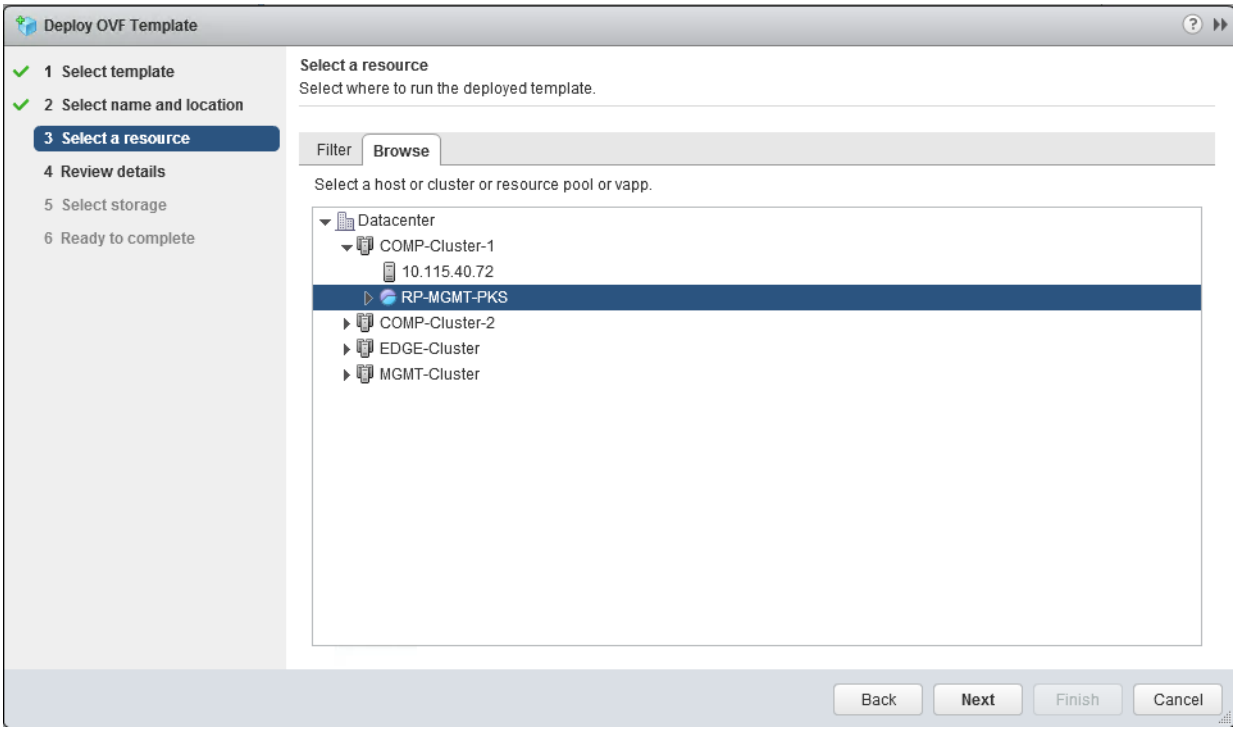
9. Review template selection and click **Next**.



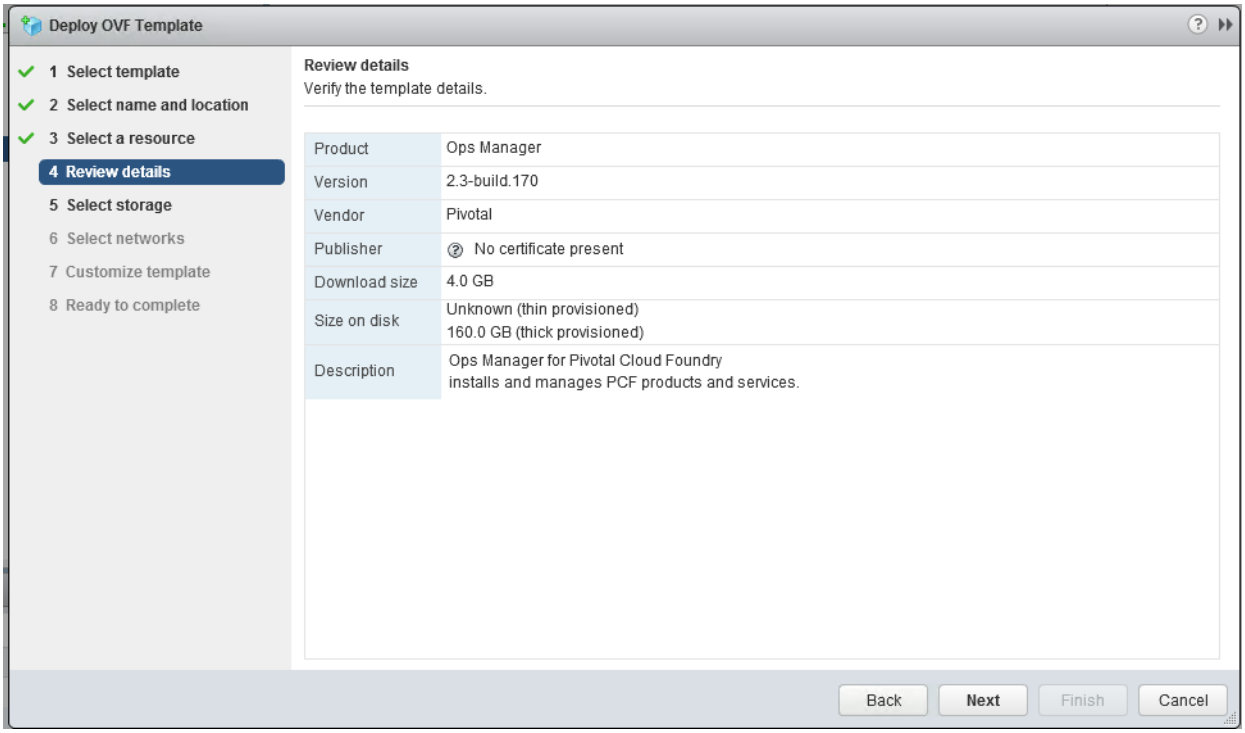
10. At the **Select Name and location** screen, enter a name for the Ops Manager VM (or use the default name), select the **Datacenter object, and click **Next**



11. At the **Select a resource** screen, select the PKS Management Plane Resource Pool and click **Next**.

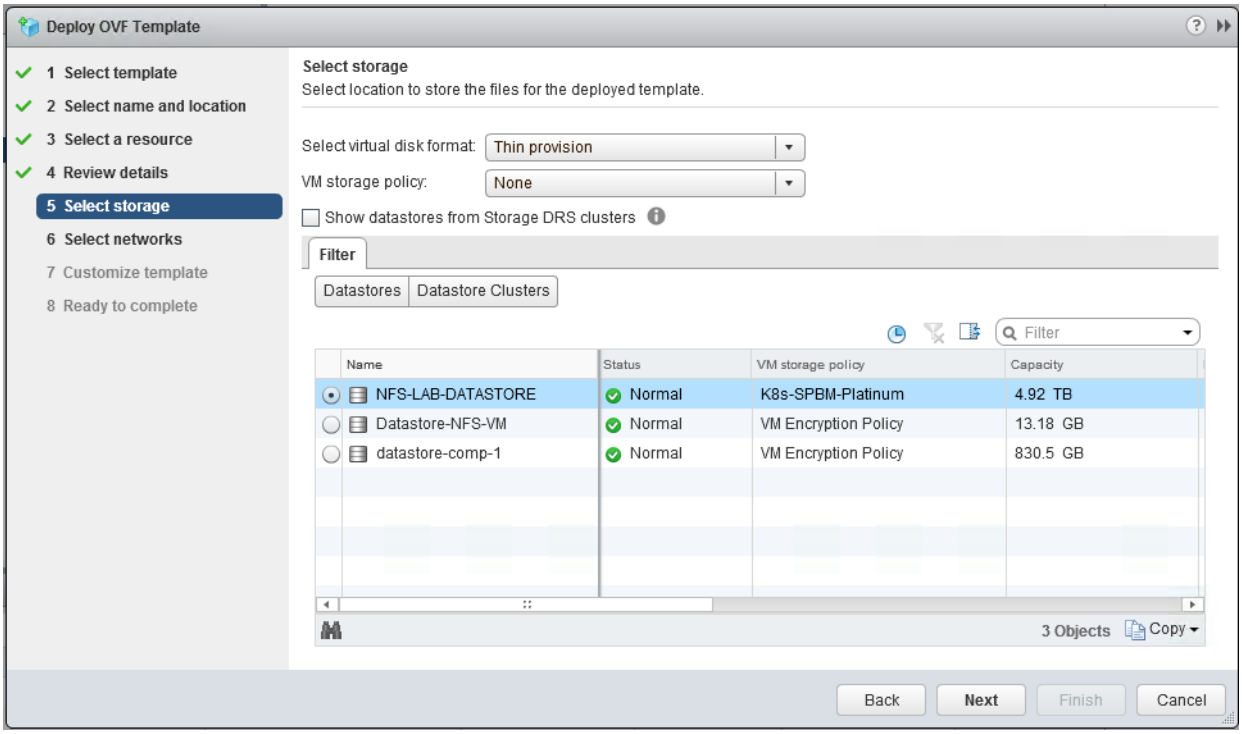


12. At the **Review Details** screen, confirm the configuration up to this point and click **Next**.

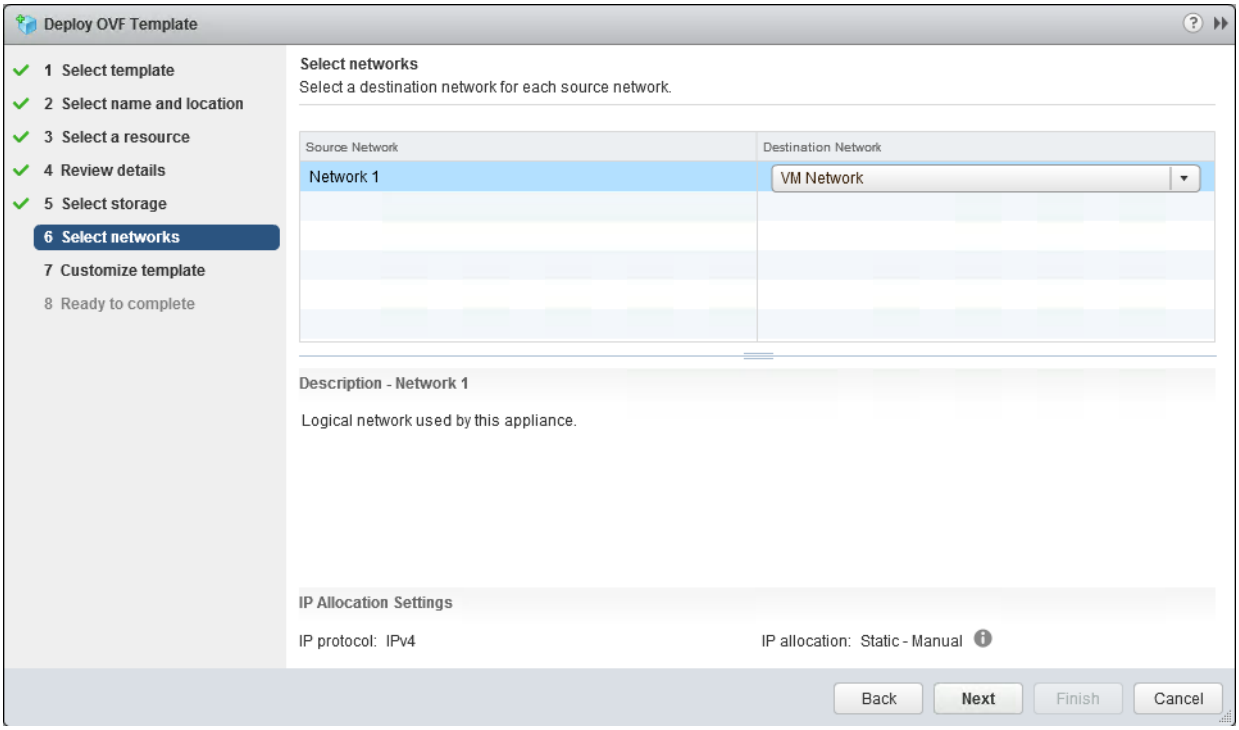


13. At the **Select Storage** screen, select **Thin Provision**, choose the desired Datastore, and click **Next**. For more information about disk formats, see Provisioning a Virtual Disk in vSphere.

> **Warning:** Ops Manager requires a Director VM with at least 8GB memory.



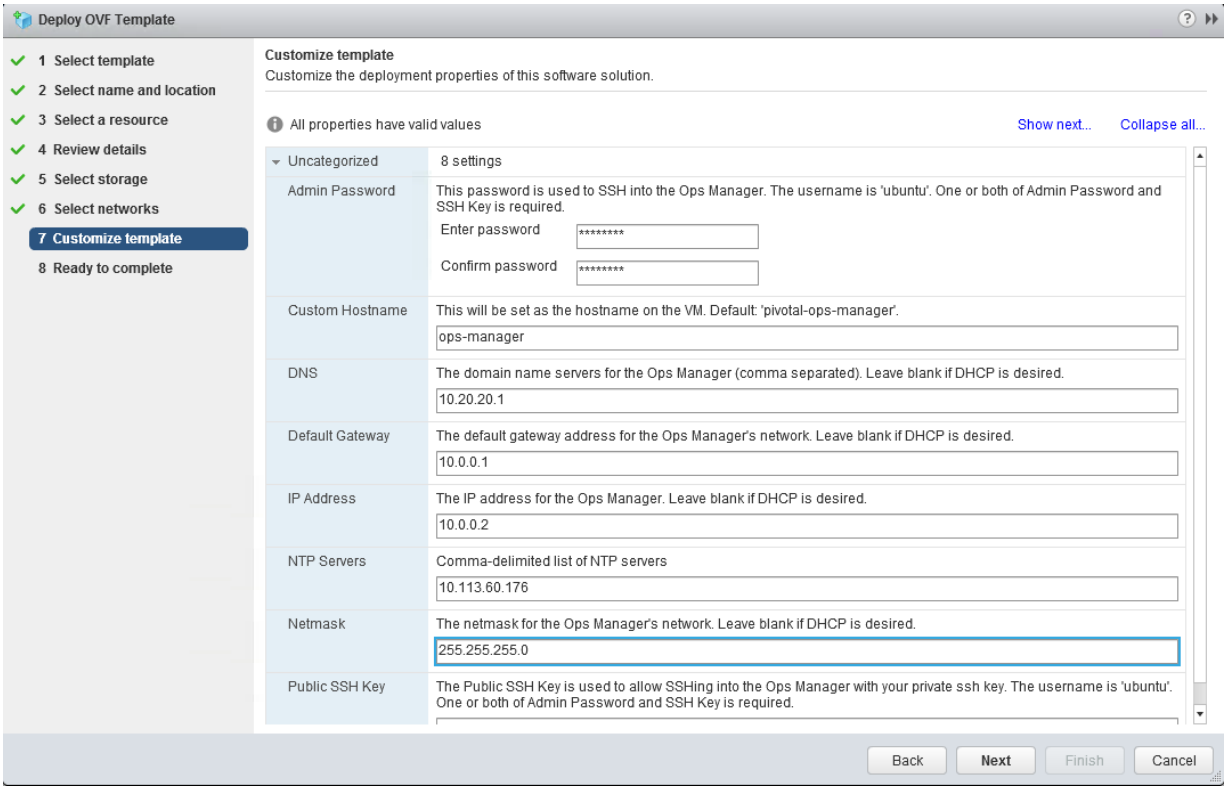
14. At the **Select Networks** screen, if you are using vSphere 6.7, select either the PKS Management T1 Logical Switch that you defined when Creating the PKS Management Plane, or if you are using vSphere 6.5, select a vSS or vDS port-group such as the standard **VM Network**, and click **Next**.



WARNING: With VMware vCenter Server 6.5, when initially deploying the Ops Manager OVA, you cannot connect to an NSX-T logical switch. You must first connect to a vSphere Standard (vSS) or vSphere Distributed Switch (vDS). After the OVA deployment is complete, before powering on the Ops Manager VM, connect the network interface to the NSX-T logical switch. The instructions below describe how to do this. This issue is resolved in VMware vCenter Server 6.7. For more information about this issue, see the VMware Knowledge Base.

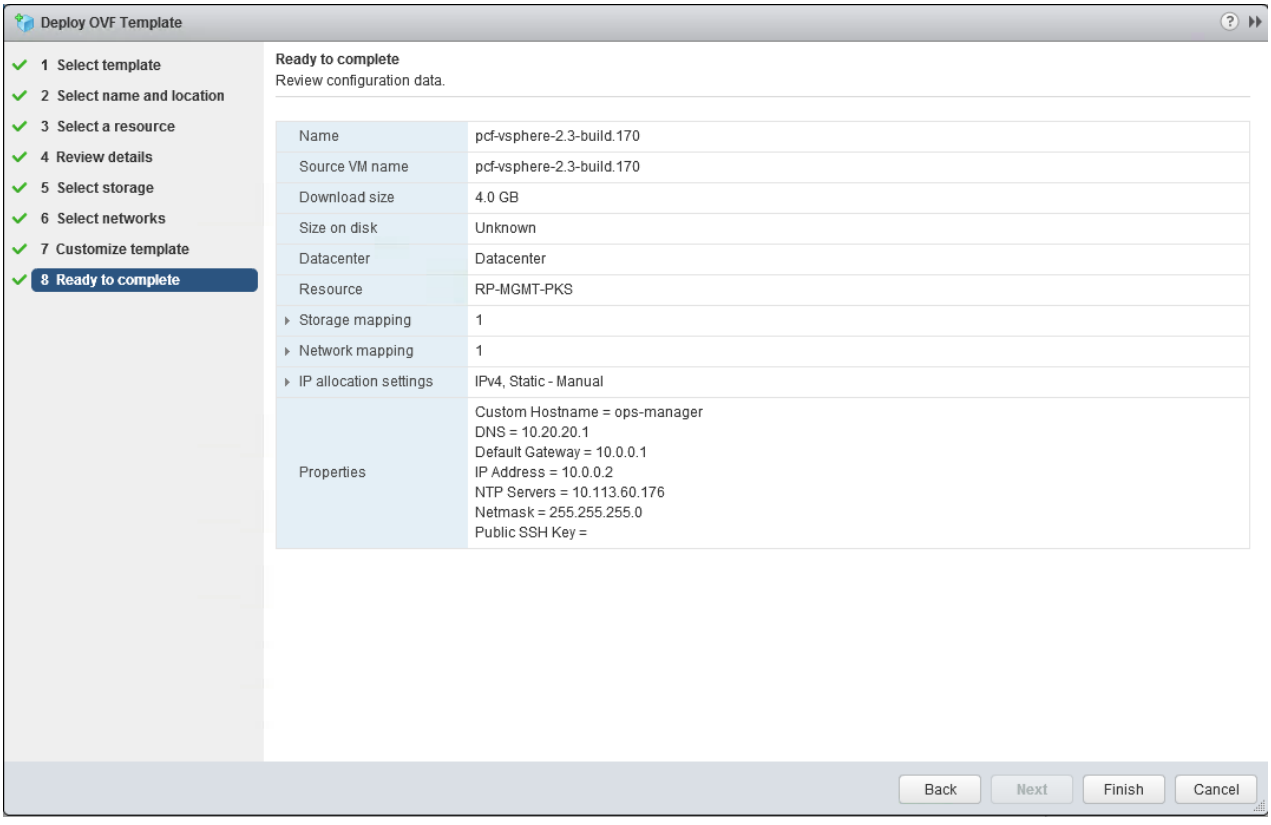
15. At the **Customize template** screen, enter the following information.

    - **Admin Password**: A default password for the “ubuntu” user. If you do not enter a password, Ops Manager will not boot up.
    - **Custom hostname**: The hostname for the Ops Manager VM, for example `ops-manager`.
    - **DNS**: One or more DNS servers for the Ops Manager VM to use, for example `10.20.20.1`.
    - **Default Gateway**: The default gateway for Ops Manager to use, for example `10.0.0.1`.
    - **IP Address**: The IP address of the Ops Manager network interface, for example `10.0.0.2` (assuming PKS NAT-mode).
    - **NTP Server**: The IP address of one or more NTP servers for Ops Manager, for example `10.113.60.176`.
    - **Netmask**: The network mask for Ops Manager, for example, `255.255.255.0`.

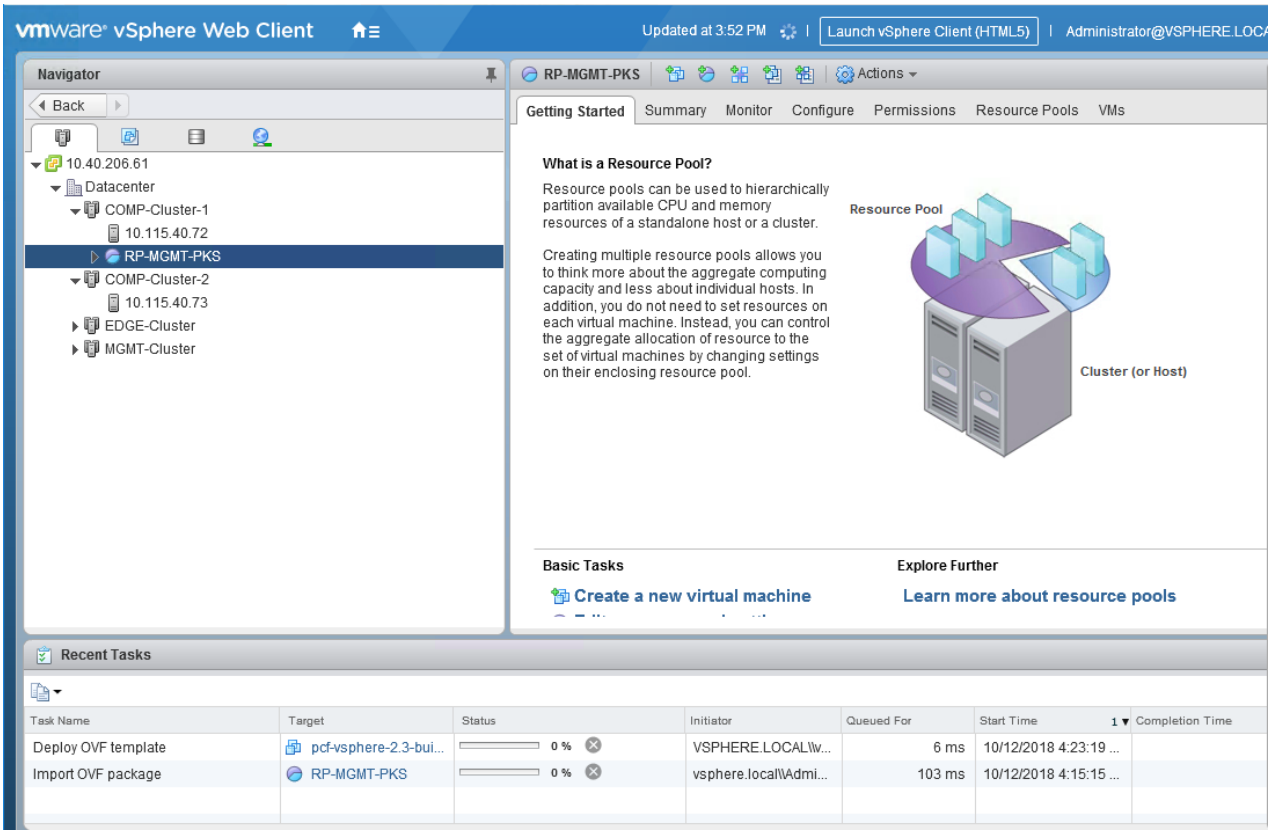


16. Click **Next**.

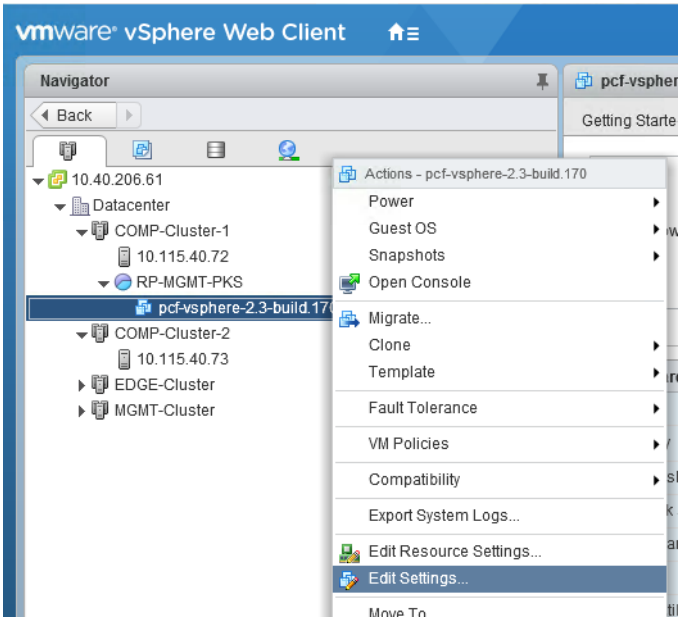
17. At the **Ready to complete** screen, review the configuration settings and click **Finish**. This action begins the OVA import and deployment process.



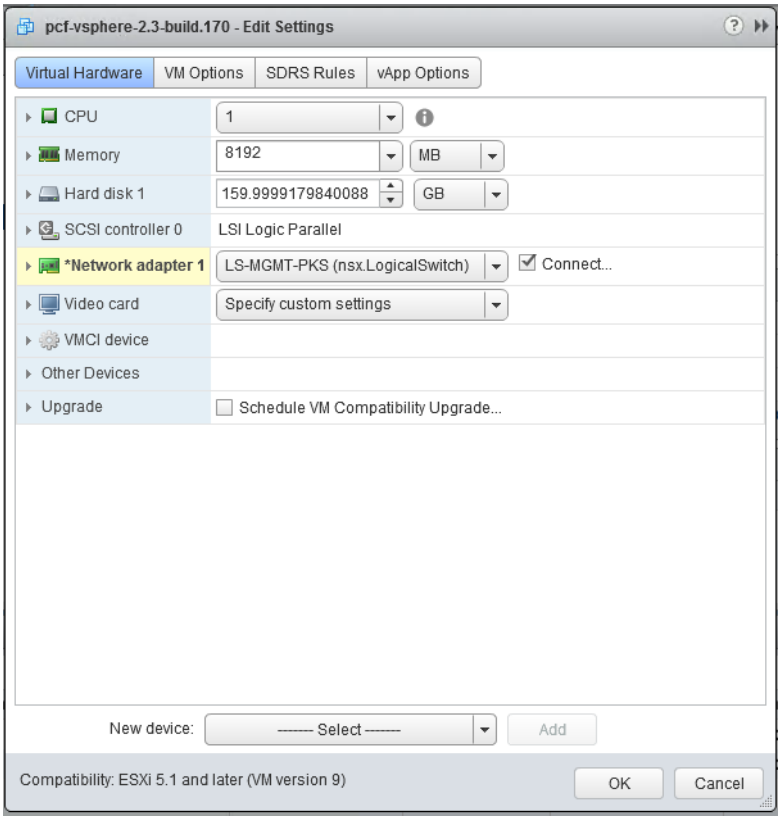
18. Use the **Recent Tasks** panel at the bottom of the vCenter dashboard to check the progress of the OVA import and deployment. IF the import or deployment is unsuccessful, check the configuration for errors.



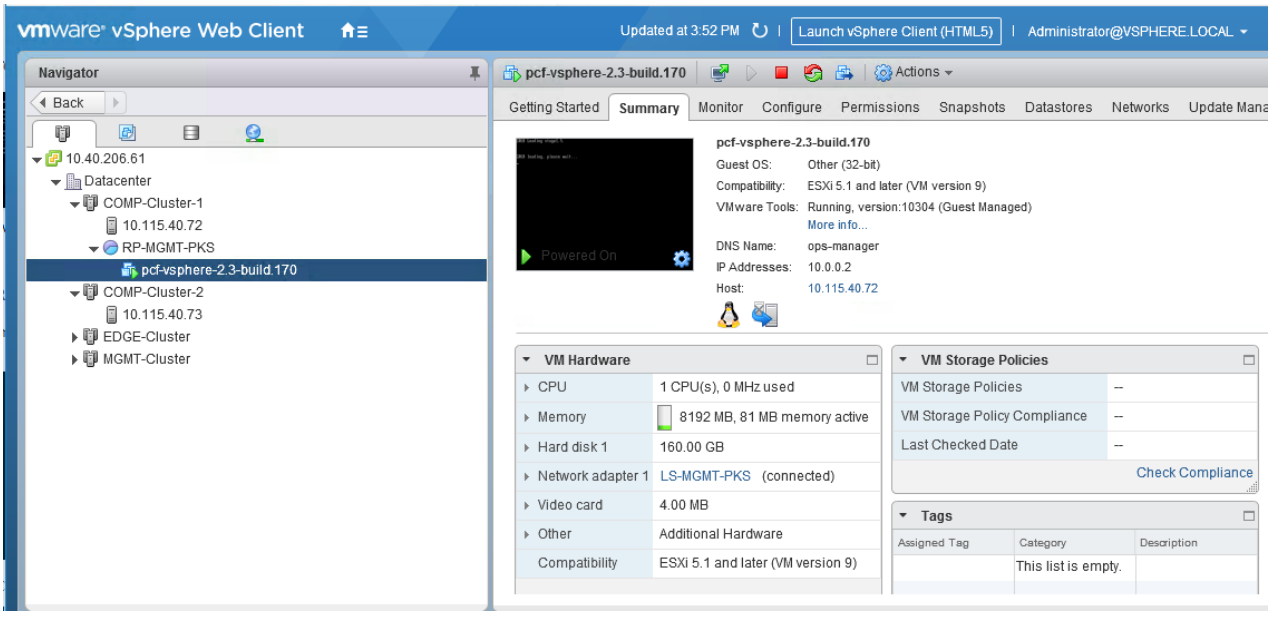
19. Once the deployment completes successfully, right-click the Ops Manager VM and select **Edit Settings**.



20. If you initially selected a vDS or vSS network for the **Virtual Hardware** > **Network adapter 1** setting, change the vNIC connection to use the `nsx.LogicalSwitch` that is defined for the PKS Management Plane, for example `LS-MGMT-PKS`. See Create PKS Management Plane if you have not defined the PKS Management T1 Logical Switch and Router.



21. Right-click the Ops Manager VM and click **Power On**.



Configure Ops Manager for PKS

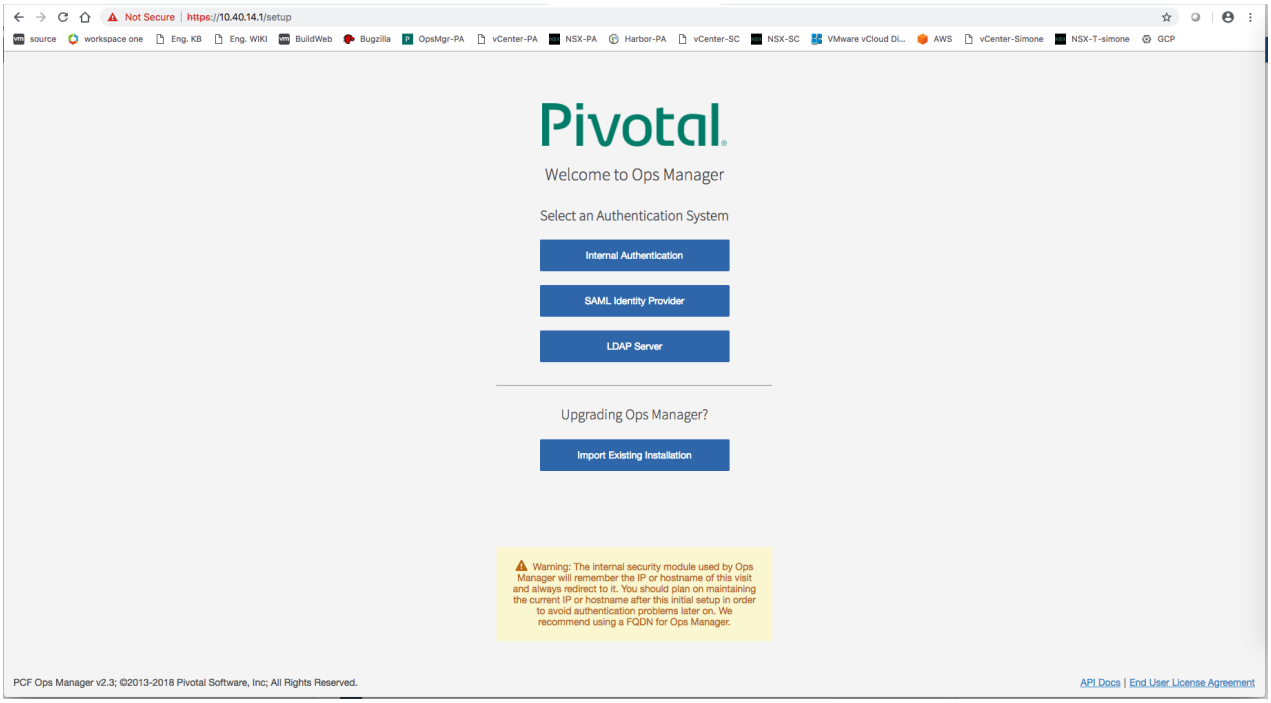
1. Create a DNS entry for the IP address that you used for Ops Manager. You must use this fully qualified domain name when you log into Ops Manager in the Installing Pivotal Cloud Foundry on vSphere topic. Use the routable IP address assigned to Ops Manager.

> **Note:** Ops Manager security features require you to create a fully qualified domain name to access Ops Manager during the initial configuration.

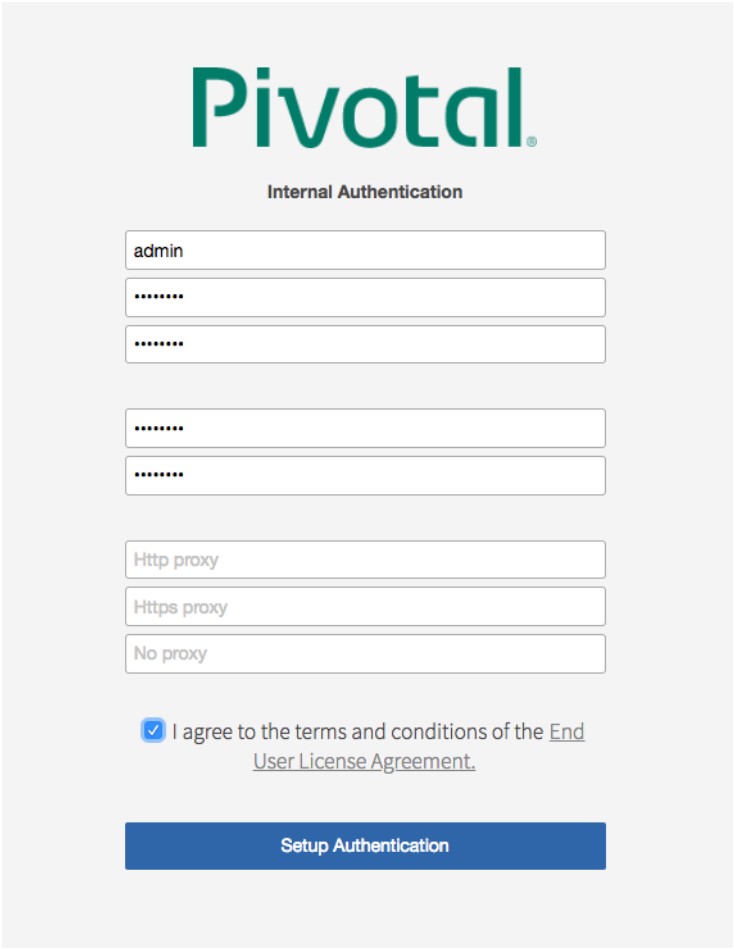
2. Navigate to the fully qualified domain of your Ops Manager in a web browser.

Note: It is normal to experience a brief delay before the interface is accessible while the web server and VM start up.

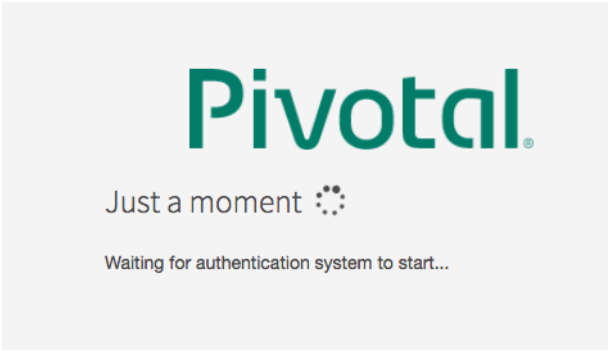
Note: If you are using the NAT deployment topology, you will need a DNAT rule that maps the Ops Manager private IP to a routable IP. See Create PKS Management Plane for instructions.



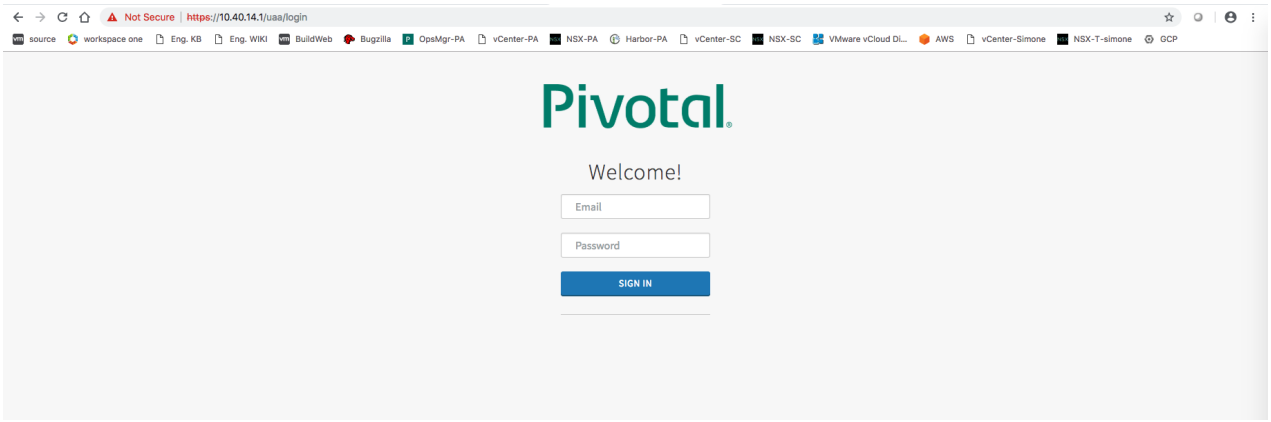
3. The first time you start Ops Manager, you are required select an authentication system. These instructions use **Internal Authentication**. See Set Up Ops Manager in the PCF documentation for configuration details for the **SAML** and **LDAP** options.



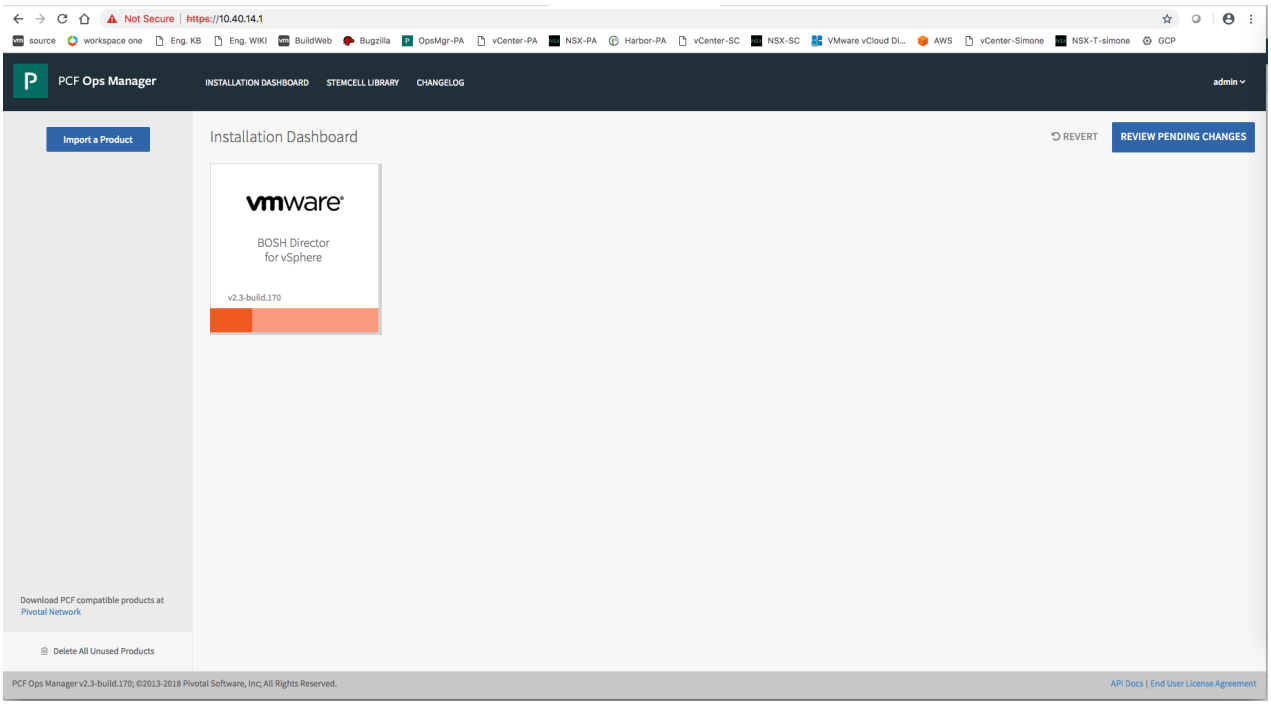
4. Select **Internal Authentication** and provide the following information:
    - **Username**, **Password**, and **Password confirmation** to create an Admin user.
    - **Decryption passphrase** and the **Decryption passphrase confirmation**. This passphrase encrypts the Ops Manager datastore, and is not recoverable.
    - **HTTP proxy** or **HTTPS proxy**, follow the instructions in Configuring Proxy Settings for the BOSH CPI.
5. Click **Setup Authentication**. It will take a few minutes to initialize the database.



6. Log in to Ops Manager with the user name and password you created.



7. Verify success. You should be able to log in, and you should see the BOSH Director tile is present and ready for configuration, indicated by the orange color.



Next Step

After you complete this procedure, follow the instructions in Generating and Registering the NSX Manager Certificate for PKS.

Please send any feedback you have to pks-feedback@pivotal.io.

Generating and Registering the NSX Manager Certificate for PKS

Page last updated:

This topic describes how to generate and register the NSX Manager certificate authority (CA) certificate in preparation for installing Pivotal Container Service (PKS) on vSphere with NSX-T.

Prerequisites

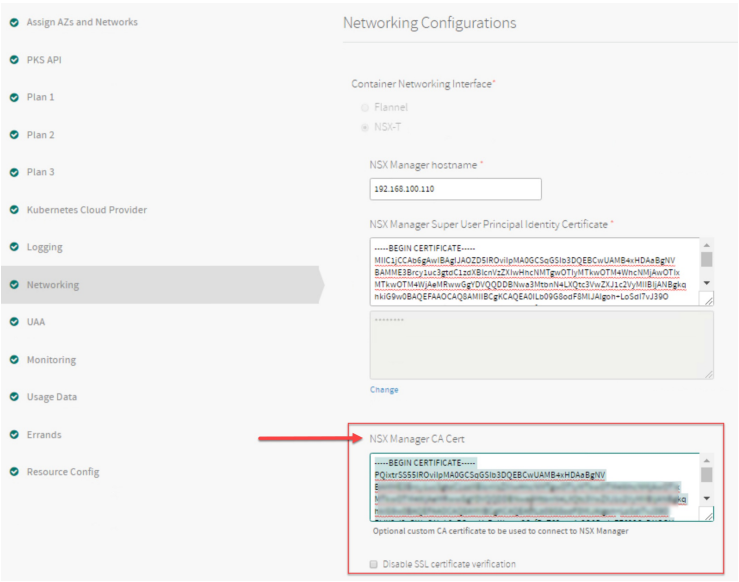
Before you begin this procedure, ensure that you have successfully completed all preceding steps for installing PKS on vSphere with NSX-T, including:

- Deploy NSX-T for PKS
- Create PKS Management Plane
- Create PKS Compute Plane
- Deploy Ops Manager with NSX-T for PKS

About the NSX Manager CA Certificate

The NSX Manager CA certificate is used to authenticate with the NSX Manager. You create an IP-based, self-signed certificate and register it with the NSX Manager. During PKS installation on vSphere with NSX-T, you provide this certificate in the **NSX Manager CA Cert** field in the **Networking** pane in the PKS tile.

See the **NSX Manager CA Cert** field in the following screenshot:



For configuration information, see the Networking section of *Installing PKS on vSphere with NSX-T*.

By default, the NSX Manager includes a self-signed API certificate with its hostname as the subject and issuer. Ops Manager requires strict certificate validation and expects the subject and issuer of the self-signed certificate to be either the IP address or fully qualified domain name (FQDN) of the NSX Manager. As a result, you need to regenerate the self-signed certificate using the FQDN of the NSX Manager in the subject and issuer field and then register the certificate with the NSX Manager using the NSX API.

The **Disable SSL certificate verification** option lets you disable validation of the NSX Manager CA certificate. Select this option for testing purposes only.

Note: The **NSX Manager CA Cert** field and the **Disable SSL certificate verification** option are intended to be mutually exclusive. If you disable SSL certificate verification, leave the CA certificate field blank. If you enter a certificate in the **NSX Manager CA Cert** field, do not disable SSL certificate verification. If you populate the certificate field and disable certificate validation, insecure mode takes precedence.

Step 1: Generate a Self-Signed CA Certificate for the NSX Manager

Complete the following steps to generate a self-signed CA certificate for the NSX Manager:

1. Create a file for the certificate request parameters named `nsx-cert.cnf`.
2. Copy the following parameters and paste them into the file, replacing `NSX-MANAGER-IP-ADDRESS` with the IP address of your NSX Manager, and `NSX-MANAGER-COMMONNAME` with the FQDN of the NSX Manager host:

```
[ req ]
default_bits = 2048
distinguished_name = req_distinguished_name
req_extensions = req_ext
prompt = no
[ req_distinguished_name ]
countryName = US
stateOrProvinceName = California
localityName = CA
organizationName = NSX
commonName = NSX-MANAGER-COMMONNAME
[ req_ext ]
subjectAltName = @alt_names
[alt_names]
DNS.1 = NSX-MANAGER-COMMONNAME,NSX-MANAGER-IP-ADDRESS
```

   For example:

```
[ req ]
default_bits = 2048
distinguished_name = req_distinguished_name
req_extensions = req_ext
prompt = no
[ req_distinguished_name ]
countryName = US
stateOrProvinceName = California
localityName = Palo-Alto
organizationName = NSX
commonName = nsxmgr-01a.example.com
[ req_ext ]
subjectAltName=DNS:nsxmgr-01a.example.com,IP:192.0.2.40
```

3. Export the `NSX_MANAGER_IP_ADDRESS` and `NSX_MANAGER_COMMONNAME` environment variables using the IP address of your NSX Manager and the FQDN of the NSX Manager host.

   For example:

```
$ export NSX_MANAGER_IP_ADDRESS=192.0.2.40
$ export NSX_MANAGER_COMMONNAME=nsxmgr-01a.example.com
```

4. Generate the certificate using openssl. Run the following command:

```
$ openssl req -newkey rsa:2048 -x509 -nodes \
-keyout nsx.key -new -out nsx.crt -subj /CN=$NSX_MANAGER_COMMONNAME \
-reqexts SAN -extensions SAN -config <(cat ./nsx-cert.cnf \
 <(printf "[SAN]\nsubjectAltName=DNS:$NSX_MANAGER_COMMONNAME,IP:$NSX_MANAGER_IP_ADDRESS")) -sha256 -days 365
```

5. Verify that the certificate looks correct and that the NSX manager IP is in the Subject Alternative Name (SAN) by running the following command:

```
$ openssl x509 -in nsx.crt -text -noout
```

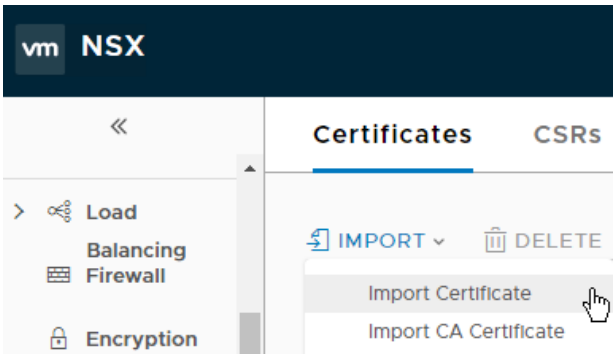
Step 2: Import the Certificate to NSX Manager

In this section you import the self-signed CA certificate you generated in the previous step to the NSX Manager.

Complete the following steps to import the certificate to the NSX Manager:

1. Log in to the NSX Manager UI.
2. Navigate to **System > Trust > Certificates**.

3. Click **Import > Import Certificate**.



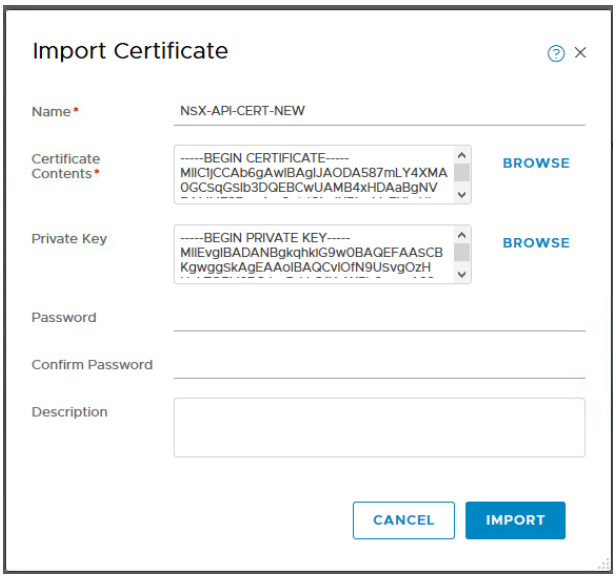
> **Note**: Make sure you select **Import Certificate** and not **Import CA Certificate**.

4. Give the certificate a unique name, such as `NSX-API-CERT-NEW`.

> **Note**: Use a unique name for the new certificate you import. The default NSX Manager CA certificate is typically named `NSX-API-CERT`.

5. Browse to and select the CA certificate and private key you generated in the previous section of steps.

6. Click **Save**.



Step 3: Register the Certificate with NSX Manager

The last step is to register the imported certificate with the NSX Manager. You must use the NSX API to register the certificate.

Complete the following steps to register the certificate with the NSX Manager:

1. To retrieve the certificate ID, run the following commands:

```
export NSX_MANAGER_IP_ADDRESS=NSX-MANAGER-IP-ADDRESS
curl --insecure -u admin:'ADMIN-PASSWORD' -X \
GET "https://$NSX_MANAGER_IP_ADDRESS/api/v1/trust-management/certificates" \
| jq -r '.results[] | select(.display_name == "CERTIFICATE-NAME") | .id'
```

   Where:

   - `NSX-MANAGER-IP-ADDRESS` is the NSX Manager IP address as determined in Step 1: Generate a Self-Signed CA Certificate for the NSX Manager.
   - `ADMIN-PASSWORD` is the administrator password.
   - `CERTIFICATE-NAME` is the certificate name.

2. To register the certificate with NSX Manager, run the following commands:

```
export NSX_MANAGER_IP_ADDRESS=NSX-MANAGER-IP-ADDRESS
export CERTIFICATE_ID="CERTIFICATE-ID" curl --insecure -u admin:'ADMIN-PASSWORD' -X \
POST "https://$NSX_MANAGER_IP_ADDRESS/api/v1/node/services/http?action=apply_certificate&certificate_id=$CERTIFICATE_ID"
```

   Where:

   - `NSX-MANAGER-IP-ADDRESS` is the NSX Manager IP address as determined in Step 1: Generate a Self-Signed CA Certificate for the NSX Manager.
   - `CERTIFICATE-ID` is the retrieved certificate ID.
   - `ADMIN-PASSWORD` is the administrator password.

Next Step

Configure BOSH Director with NSX-T for PKS.

Configuring BOSH Director with NSX-T for PKS

Page last updated:

This topic describes how to configure BOSH Director for vSphere with NSX-T integration for PKS.

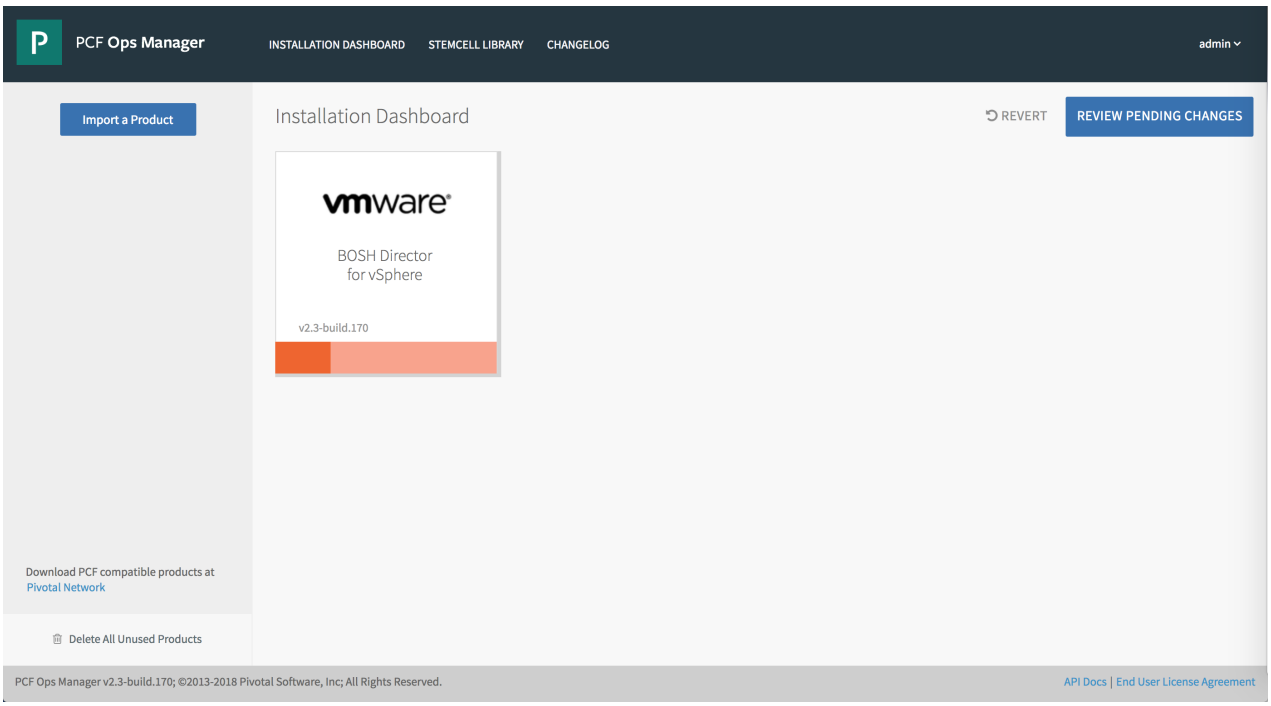
Prerequisites

Before you begin this procedure, ensure that you have successfully completed all preceding steps for installing PKS on vSphere with NSX-T, including:

- Deploying NSX-T for PKS
- Creating the PKS Management Plane
- Creating the PKS Compute Plane
- Deploying Ops Manager with NSX-T for PKS
- Generating and Registering the NSX Manager Certificate for PKS

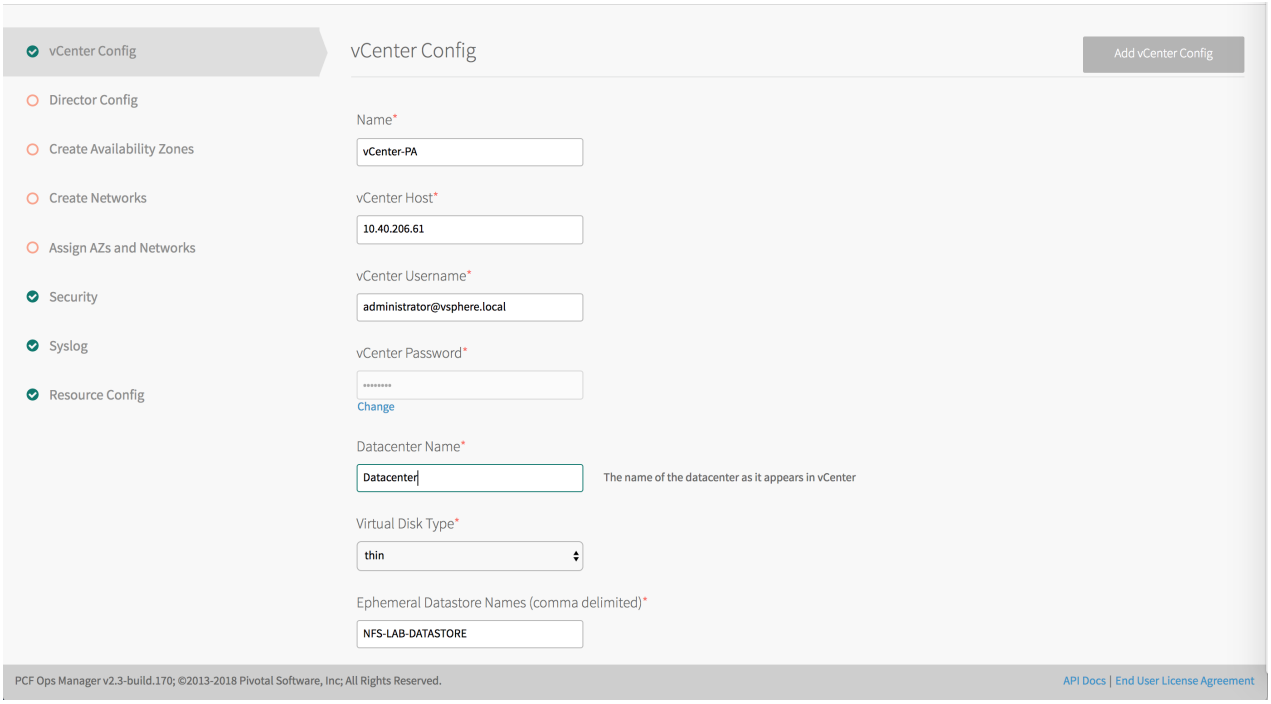
Step 1: Log in to Ops Manager

1. Log in to Ops Manager with the Admin username and password credentials.
2. Click the **BOSH Director for vSphere** tile.



Step 2: Configure vCenter for PKS

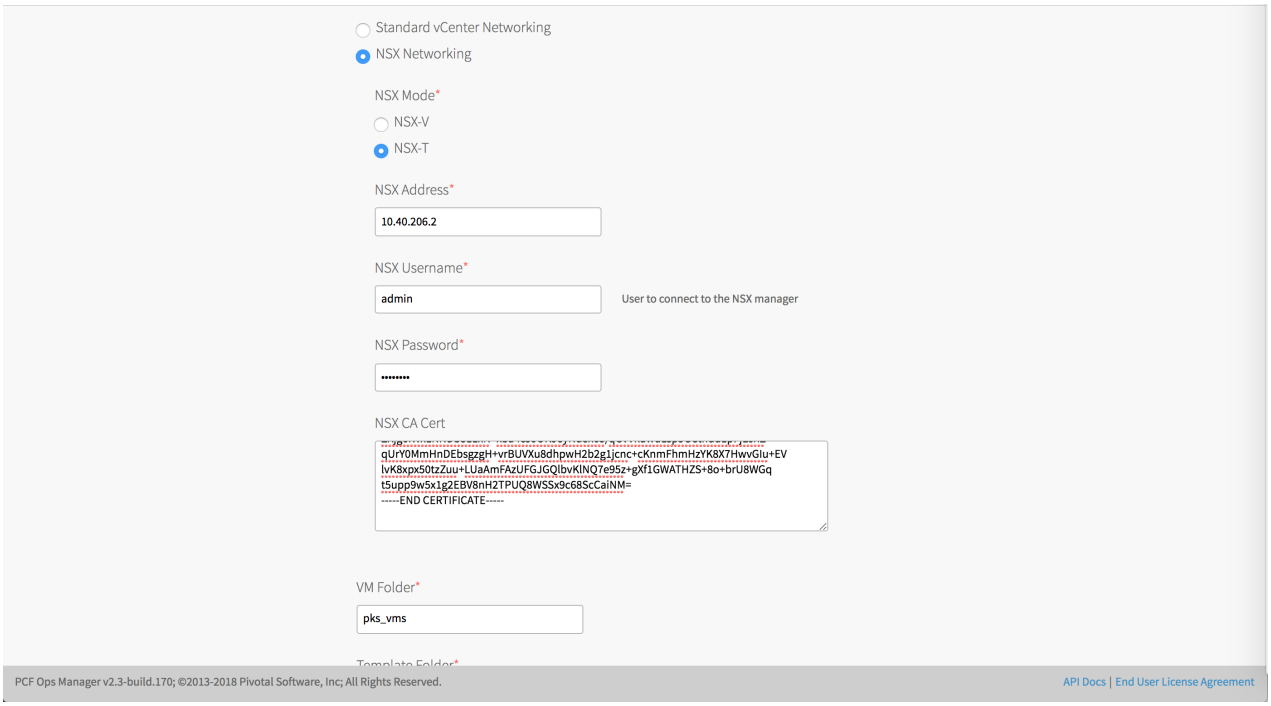
1. Select **vCenter Config**.



2. Enter the following information:

    - **vCenter Host**: The hostname of the vCenter that manages ESXi/vSphere.
    - **vCenter Username**: A vCenter username with create and delete privileges for virtual machines (VMs) and folders.
    - **vCenter Password**: The password for the vCenter user specified above.
    - **Datacenter Name**: The name of the datacenter as it appears in vCenter.
    - **Virtual Disk Type**: The Virtual Disk Type to provision for all VMs. For guidance on selecting a virtual disk type, 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.

3. Select **NSX Networking**, then select **NSX-T**.



4. Configure NSX-T networking as follows:

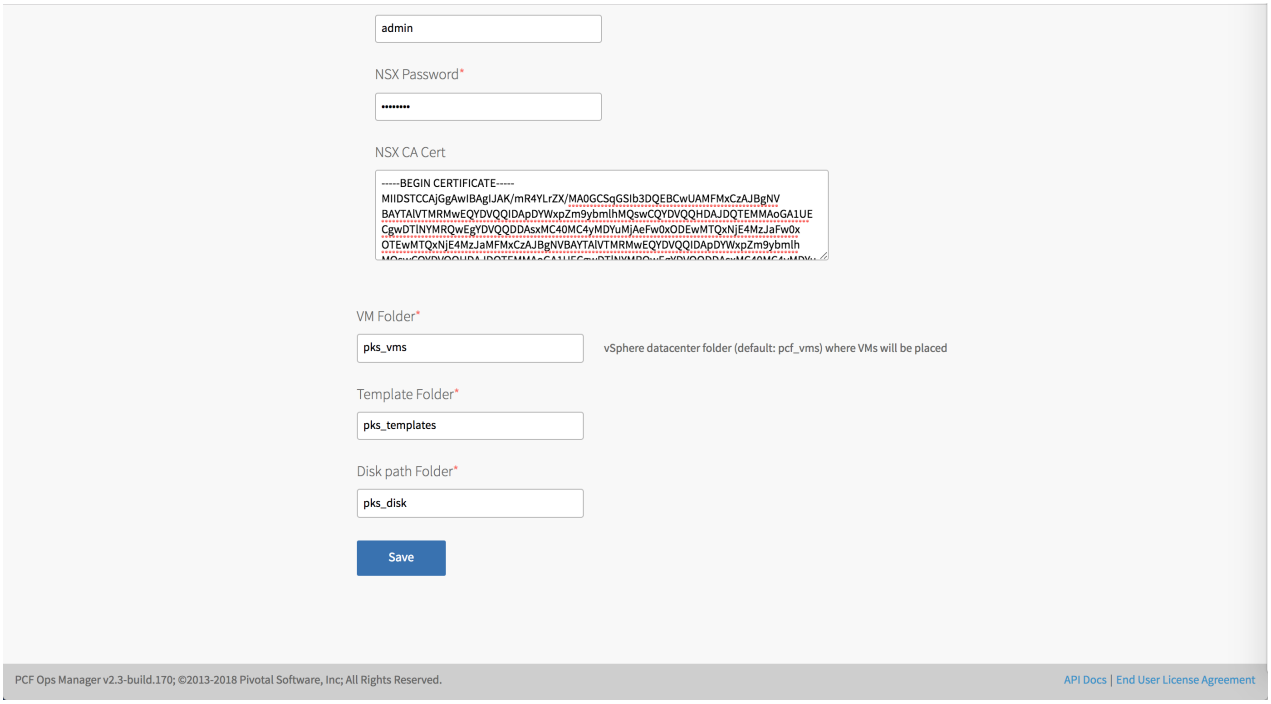
    - **NSX Address**: Enter the IP address of the NSX Manager host.
    - **NSX Username** and **NSX Password**: Enter the NSX Manager username and password.

- **NSX CA Cert**: Provide the CA certificate in PEM format that authenticates to the NSX server. Open the NSX CA Cert that you generated and copy/paste its content to this field.

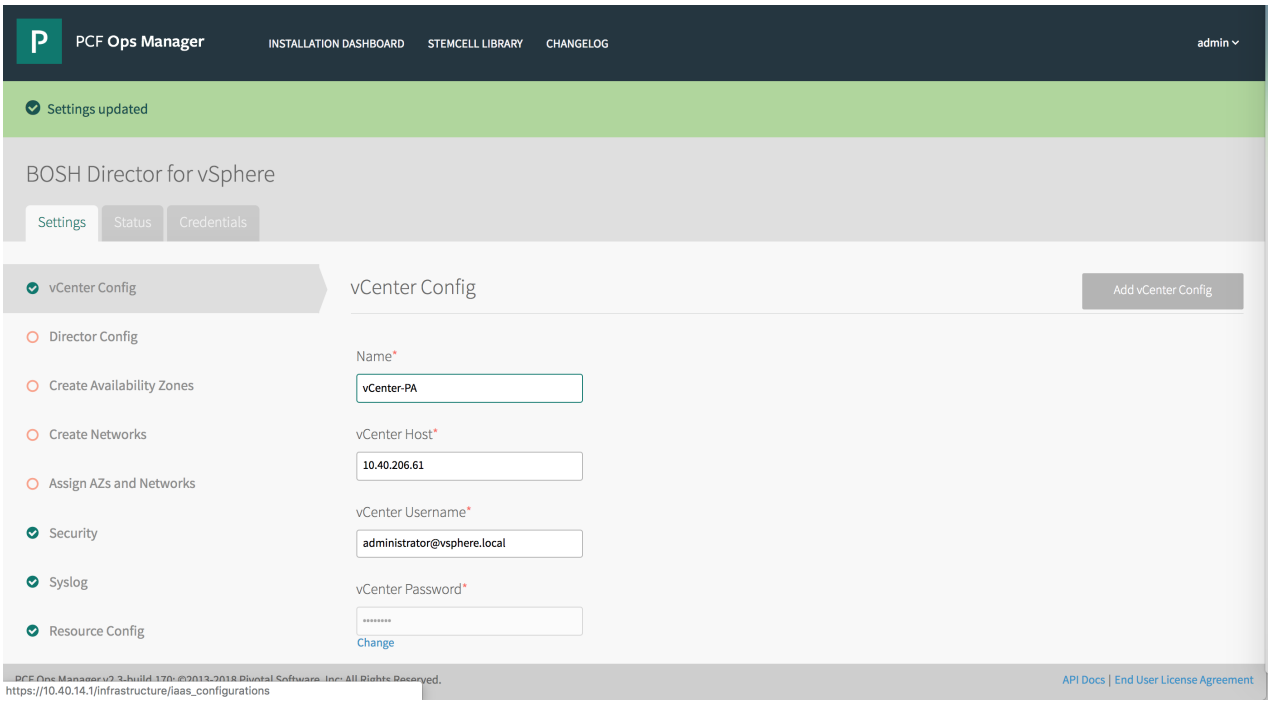
5. Configure the following folder names:

   - **VM Folder**: The vSphere datacenter folder where Ops Manager places VMs. Enter `pks_vms`.
   - **Template Folder**: The vSphere datacenter folder where Ops Manager places VMs. Enter `pks_templates`.
   - **Disk path Folder**: The vSphere datastore folder where Ops Manager creates attached disk images. You must not nest this folder. Enter `pks_disk`.

> **Note**: After your initial deployment, you cannot edit the VM Folder, Template Folder, and Disk path Folder names.

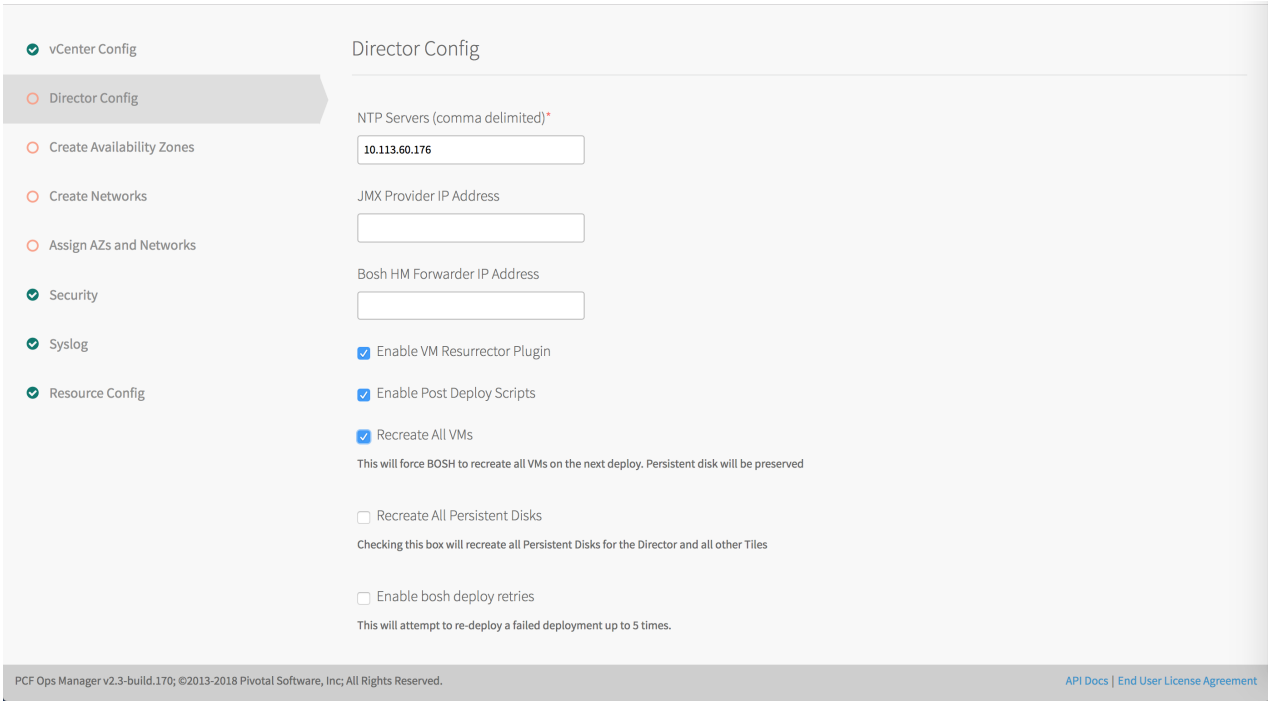


6. Click **Save**.



Step 3: Configure BOSH Director

1. Select **Director Config**.



2. In the **NTP Servers (comma delimited)** field, enter your NTP server addresses.

> **Note:** The NTP server configuration only updates after VM recreation. Ensure that you select the **Recreate all VMs** checkbox if you modify the value of this field.

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

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

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

> **Note:** Starting in PCF v2.0, BOSH-reported component metrics are available in the Loggregator Firehose by default. If you continue to use the BOSH HM Forwarder to consume these component metrics, you may receive duplicate data. To prevent this, leave the **Bosh HM Forwarder IP Address** field blank. For additional guidance, see BOSH System Metrics Available in Loggregator Firehose in the PCF v2.0 Release Notes.

5. Select the **Enable VM Resurrector Plugin** to enable BOSH Resurrector functionality.

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

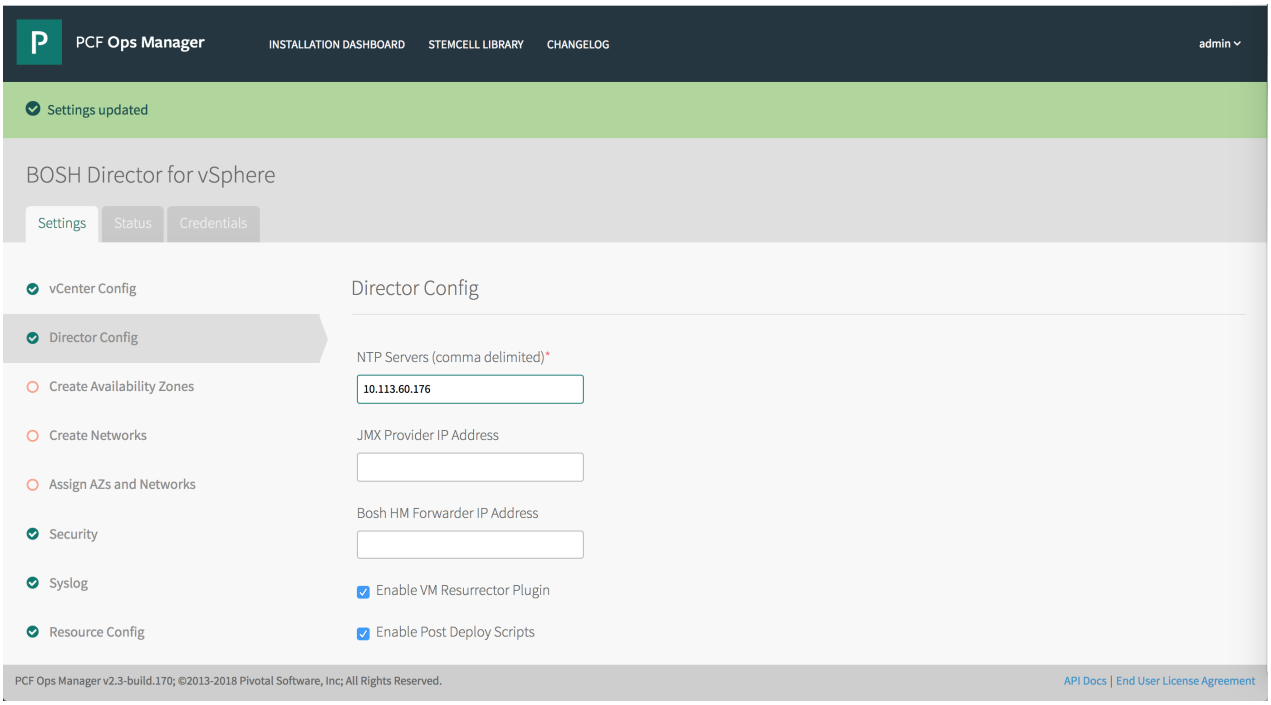
> **Note**: You must enable post-deploy scripts to install PKS.

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

8. For typical PKS deployments, the default settings for all other BOSH Director configuration parameters are suitable. Optionally you can apply additional configurations to BOSH Director. See Director Config Page in *Configuring BOSH Director on vSphere* in the PCF documentation for details.

> **Note**: If you need to be able to remotely access the BOSH Director VM using the BOSH CLI, and you are deploying PKS with NSX-T in a NAT topology, you must provide the **Director Hostname** for BOSH at the time of installation. See Director Config Page in *Configuring BOSH Director on vSphere* in the PCF documentation for details.

9. Click **Save**.

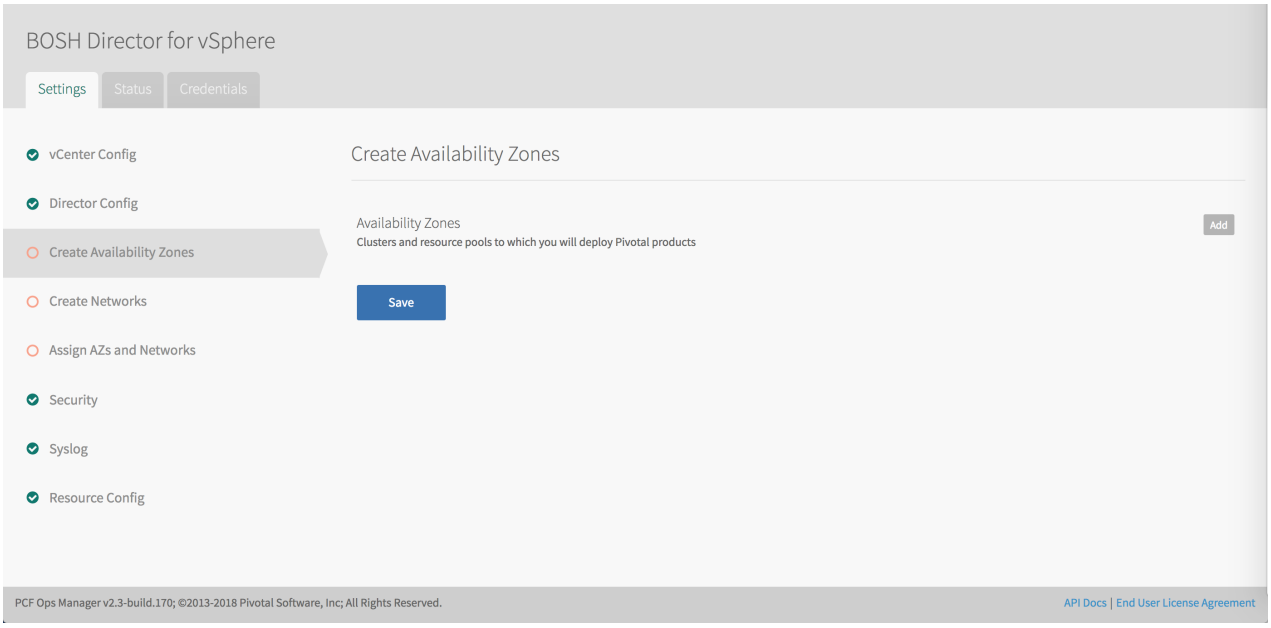


Step 4: Create Availability Zones

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 your chosen runtime.

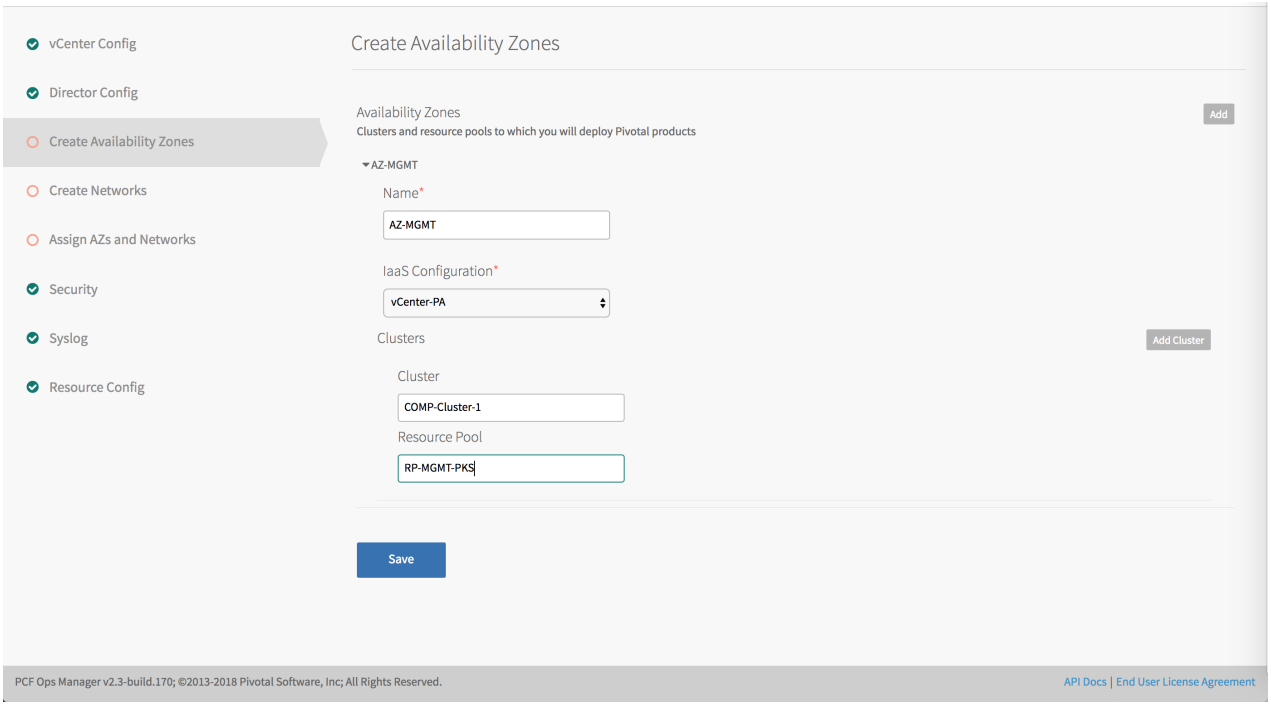
Note: For more information about using availability zones in vSphere, see Understanding Availability Zones in VMware Installations in the PCF documentation.

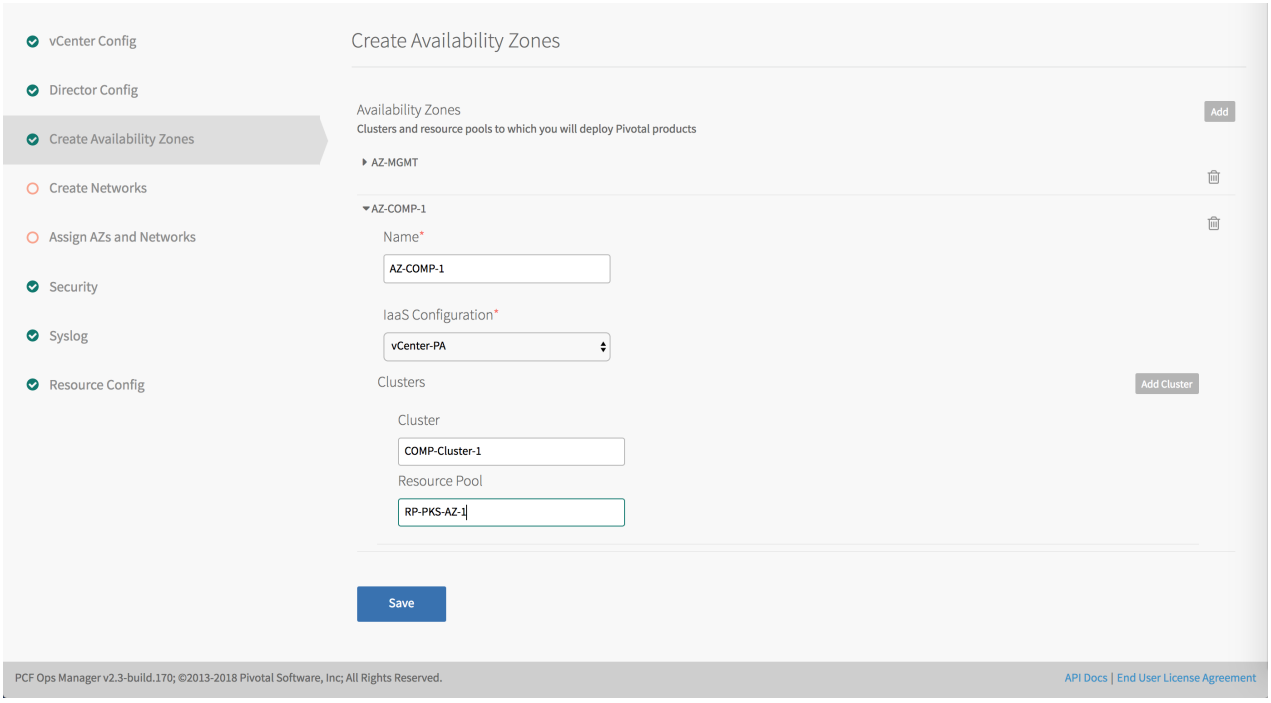
1. Select **Create Availability Zones**.

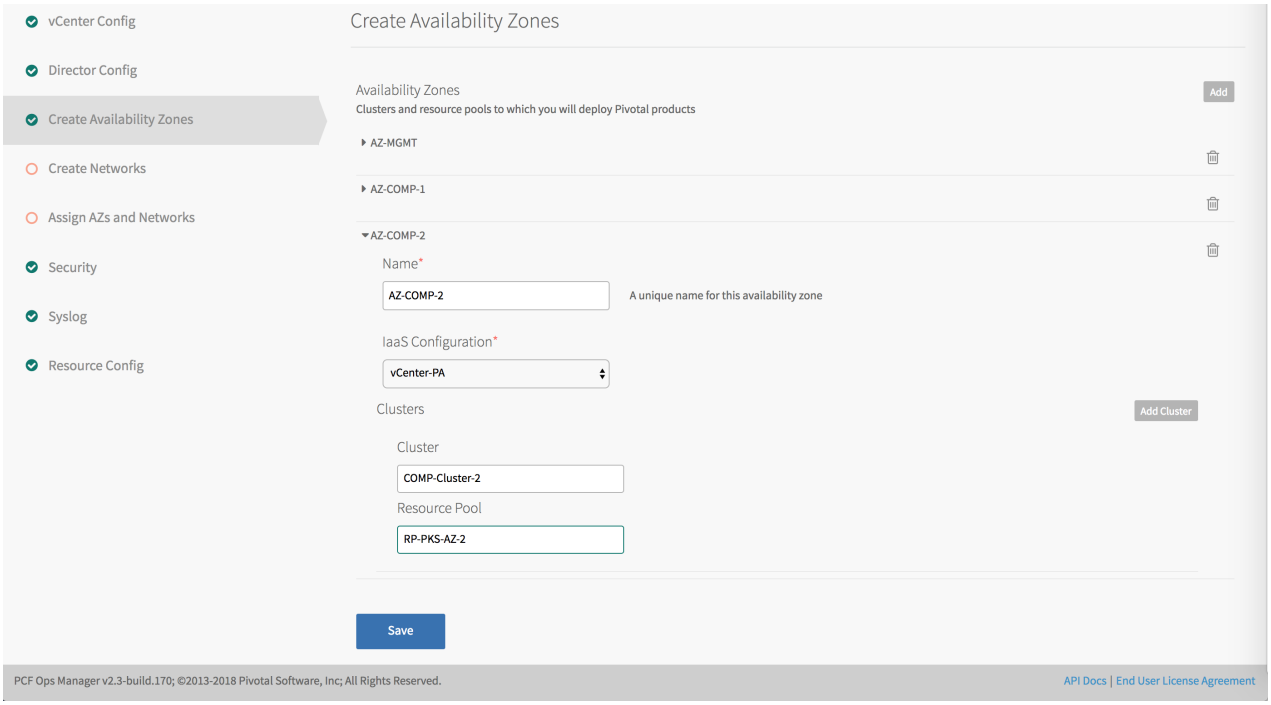


2. Use the following steps to create one or more Availability Zones for PKS to use:
    - Click **Add** and create the PKS Management AZ.
    - Enter a unique **Name** for the Availability Zone, such as `AZ-MGMT`.

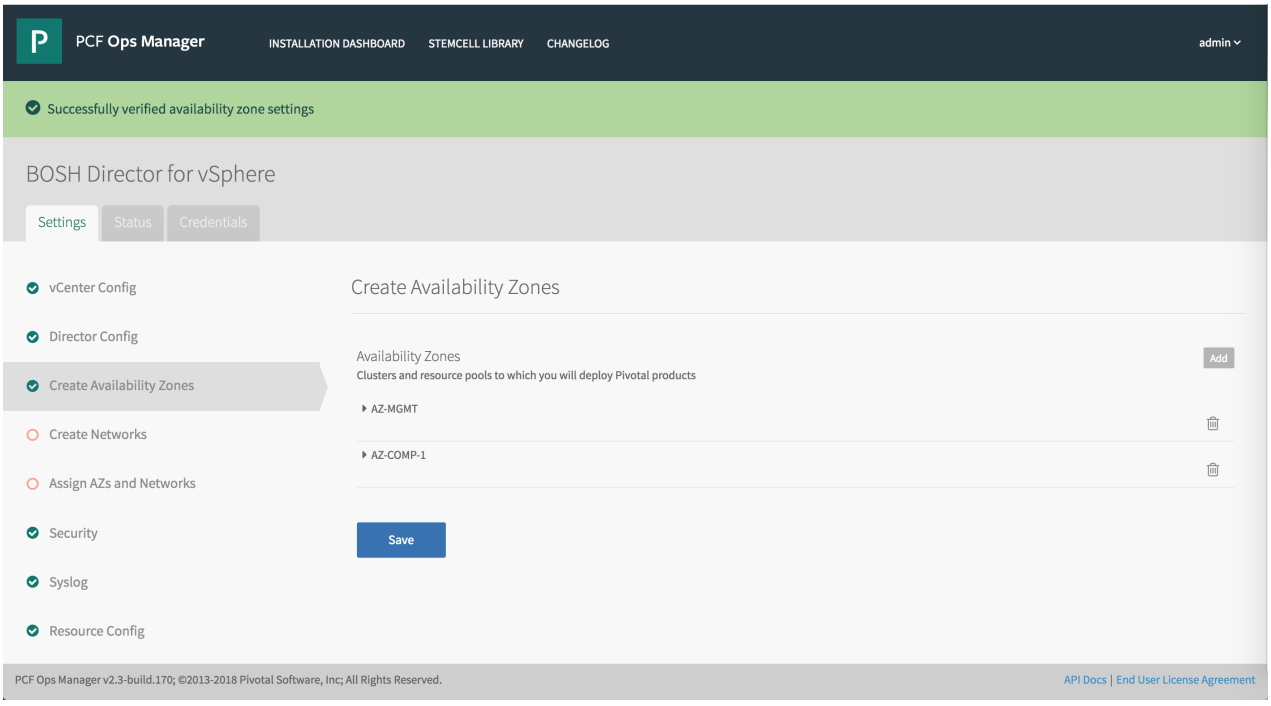
- Select the IaaS configuration (vSphere/vCenter).
- Enter the name of an existing vCenter **Cluster** to use as an Availability Zone, such as `COMP-Cluster-1`.
- Enter the name of the **PKS Management Resource Pool** in the vCenter cluster that you specified above, such as `RP-MGMT-PKS`. The jobs running in this Availability Zone share the CPU and memory resources defined by the pool.
- Click **Add Cluster** and create at least one PKS Compute AZ.
- Specify the **Cluster** and the **Resource Pool**, such as `RP-PKS-AZ`.
- Add additional clusters as necessary. Click the trash icon to delete a cluster. The first cluster cannot be deleted.





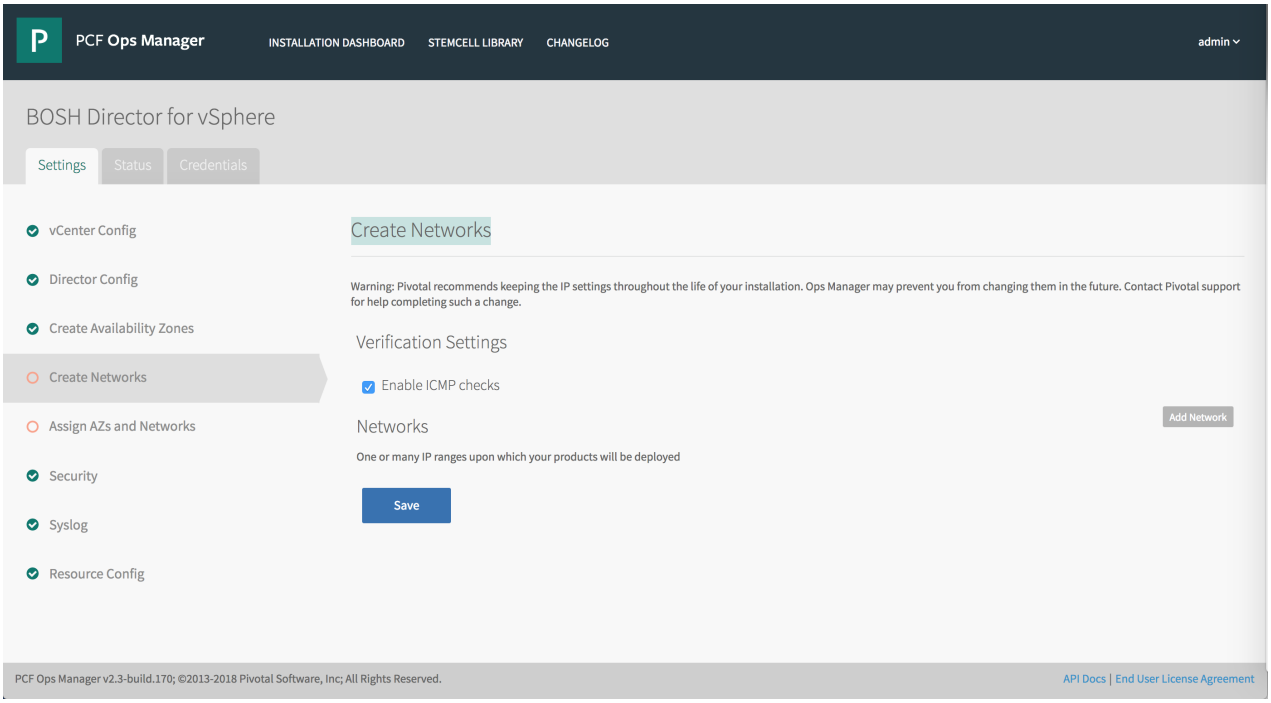


3. Click **Save**.



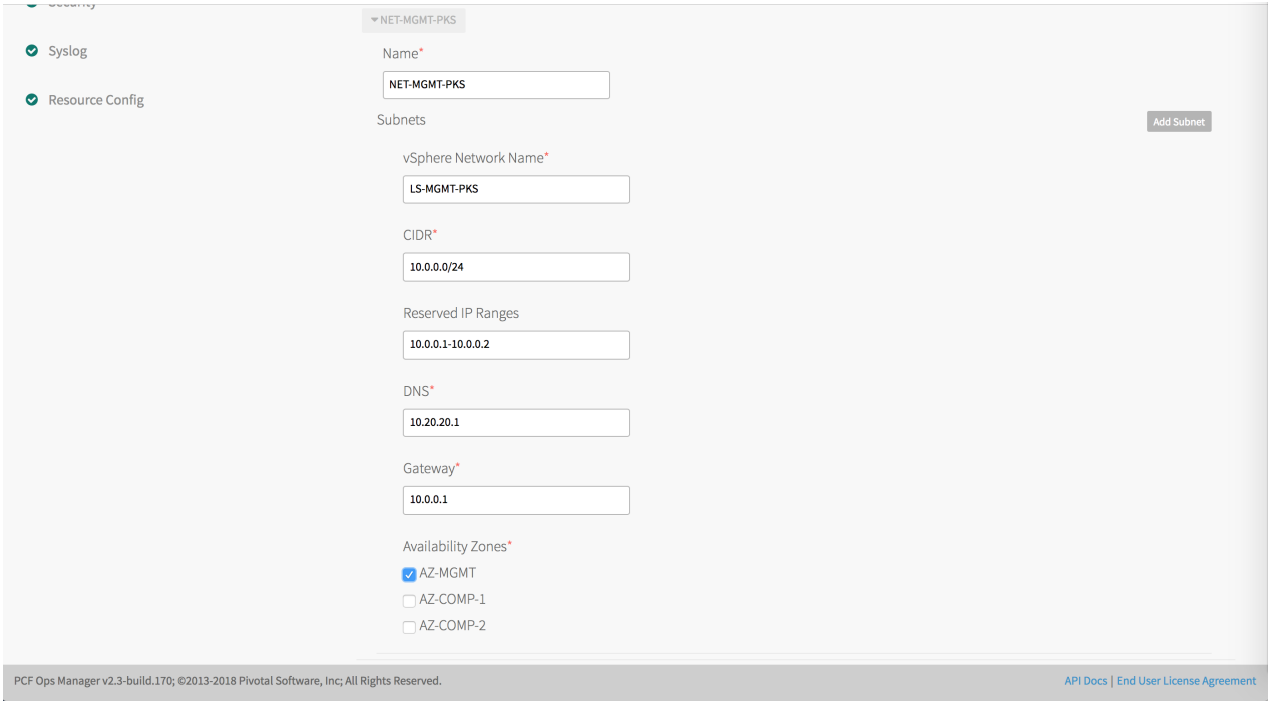
Step 5: Create Networks

1. Select **Create Networks**.



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

3. Click **Add Network**.



4. Create the following network:

    - `NET-MGMT-PKS`: Network for Ops Manager, BOSH Director, and the PKS API. This network maps to the NSX logical switch created for the PKS Management Network. See Creating PKS Management Plane.

> **Note**: NSX-T automatically creates the service network to be used by the master and worker nodes (VMs) for Kubernetes clusters managed by PKS. You should not manually create this network.

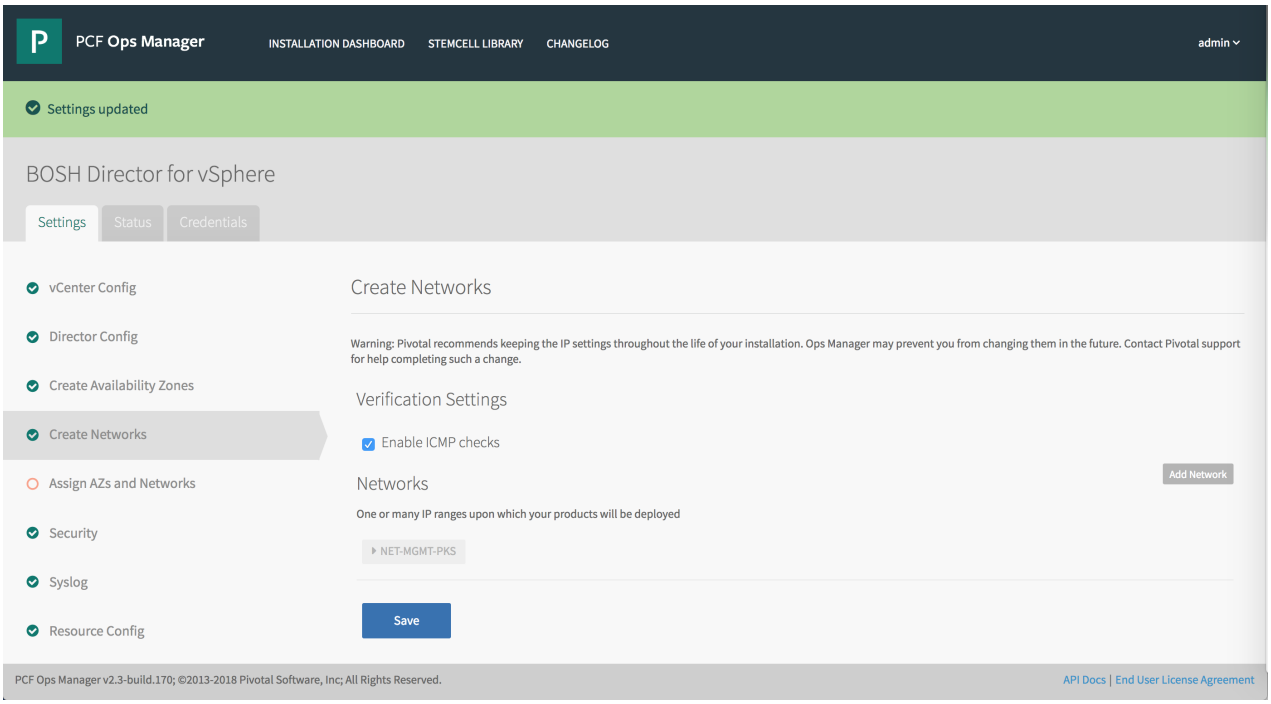
Use the following values as a guide when you define the network in BOSH. Replace the IP addresses with ranges you defined for the PKS Management Network.. Reserve any IP addresses from the subnet that are already in use, such as the IP for Ops Manager and subnet gateway.

	Field	Configuration
Infrastructure Network	Name	`NET-MGMT-PKS`
	vSphere Network Name	`LS-MGMT-PKS`
	CIDR	`10.0.0.0/24`
	Reserved IP Ranges	`10.0.0.1-10.0.0.2`
	DNS	`10.20.20.1`
	Gateway	`10.0.0.1`

1. Select the **AZ-MGMT** Availability Zone** to use with the NET-MGMT-PKS network.

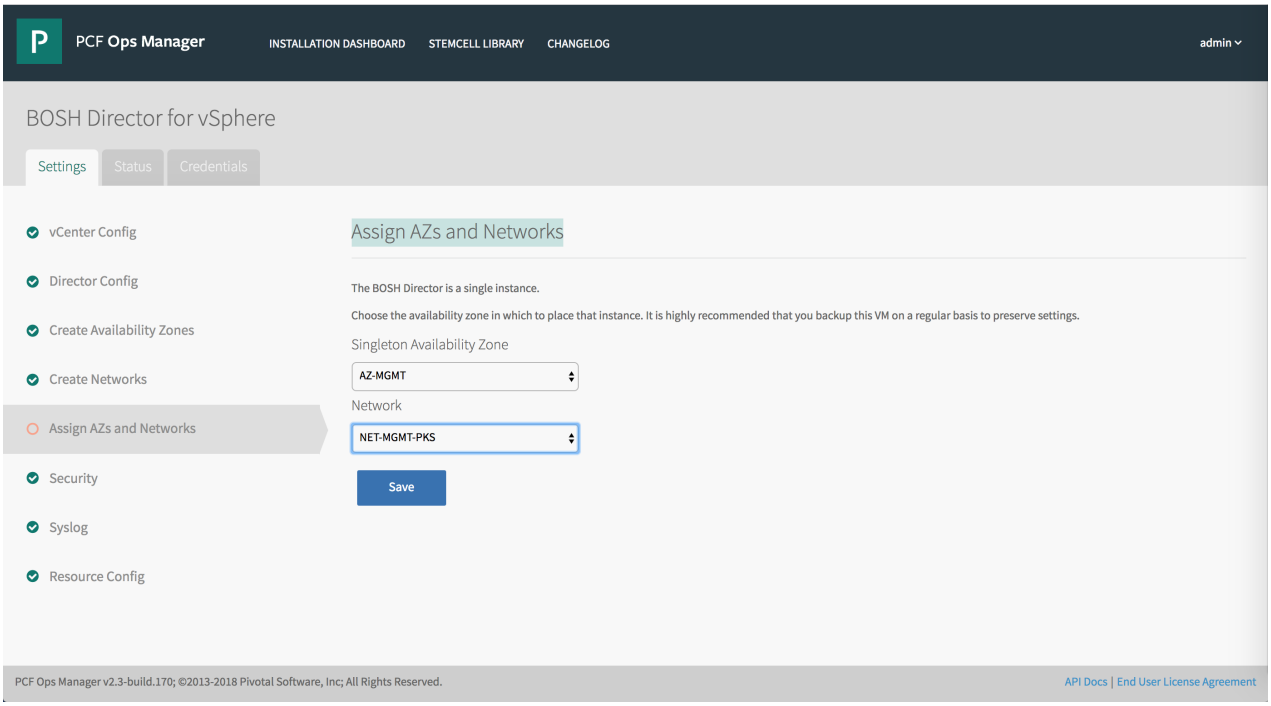
> **Note**: Do not select the COMPUTE network at this point in the configuration. It will be performed at the end of the procedure.

2. Click **Save**.



Step 6: Assign AZs and Networks

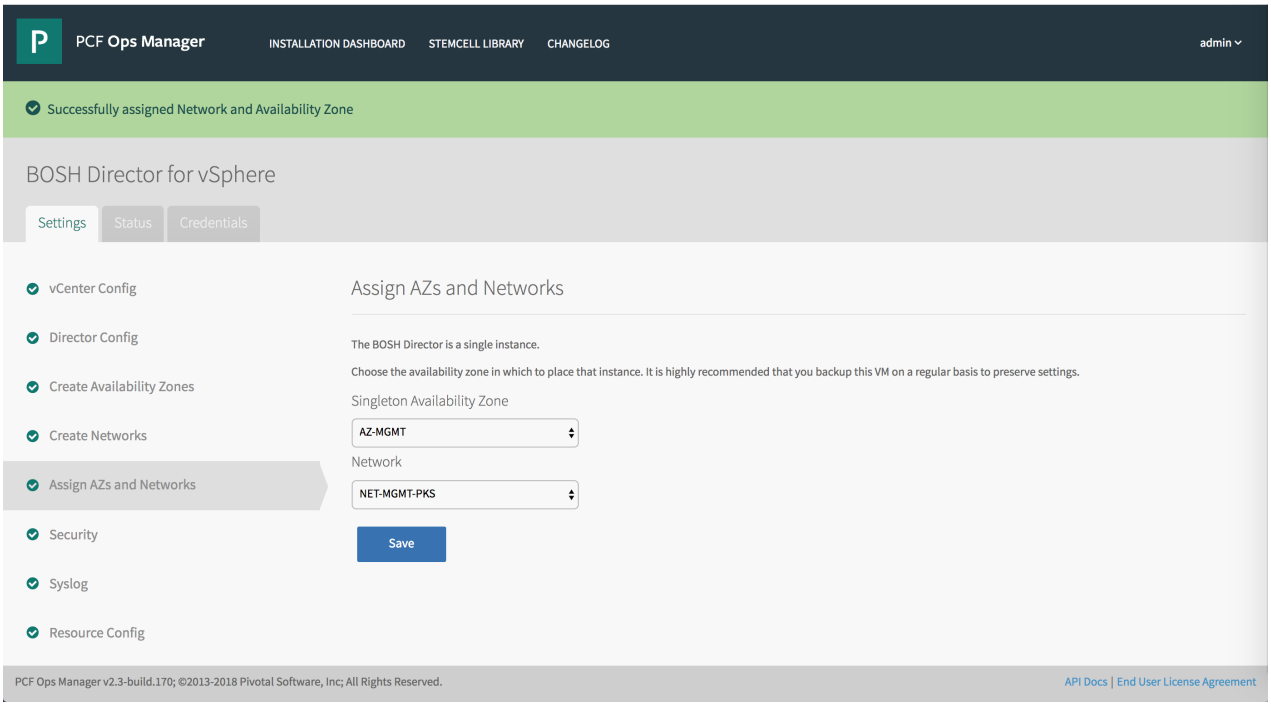
1. Select **Assign AZs and Networks**.



2. Use the drop-down menu to select a **Singleton Availability Zone**. The Ops Manager Director installs in this Availability Zone. For PKS, this will be the `AZ-MGMT` availability zone.

3. Use the drop-down menu to select a **Network** for BOSH Director. BOSH Director runs on the PKS Management Plane network. Select the `NST-MGTM-PKS` network.

4. Click **Save**.



Step 7: Configure Security

1. Select **Security**.

2. In **Trusted Certificates**, enter a custom certificate authority (CA) certificate to insert into your organization's certificate trust chain. This allows all BOSH-deployed components in your deployment to trust a custom root certificate. If you are using a private Docker registry, such as VMware

Harbor, use this field to enter the certificate for the registry. See Integrating Harbor Registry with PKS for details.

3. Choose **Generate passwords** or **Use default BOSH password**. Pivotal recommends that you use the **Generate passwords** option for increased security.
4. Click **Save**. To view your saved Director password, click the **Credentials** tab.

Step 8: Configure Logging

1. Select **Syslog**.
2. (Optional) To send BOSH Director system logs to a remote server, select **Yes**.
3. In the **Address** field, enter the IP address or DNS name for the remote server.
4. In the **Port** field, enter the port number that the remote server listens on.
5. In the **Transport Protocol** dropdown menu, select **TCP** or **UDP**. This selection determines which transport protocol is used to send the logs to the remote server.
6. (Optional) Mark the **Enable TLS** checkbox to use TLS encryption when sending logs to the remote server.
    - In the **Permitted Peer** field, enter either the name or SHA1 fingerprint of the remote peer.
    - In the **SSL Certificate** field, enter the SSL certificate for the remote server.
7. Click **Save**.

Step 9: Configure Resources

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

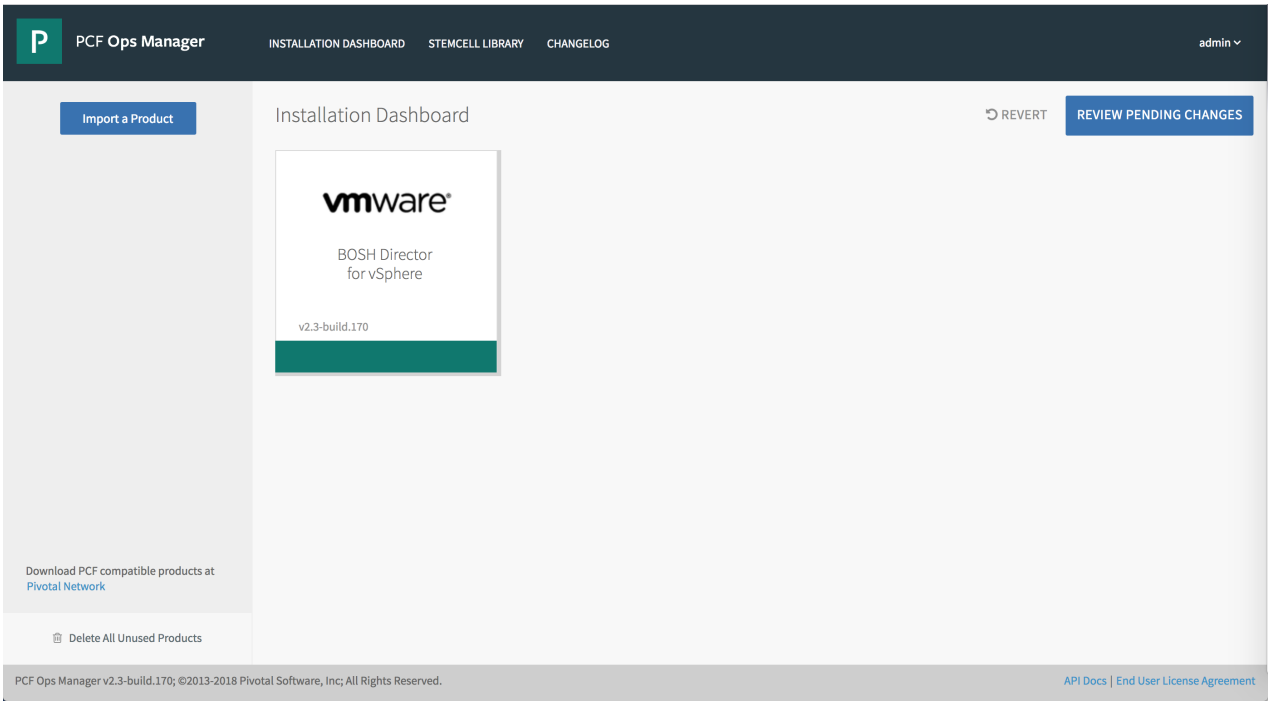
    Note: Ops Manager requires a Director VM with at least 8 GB memory.

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

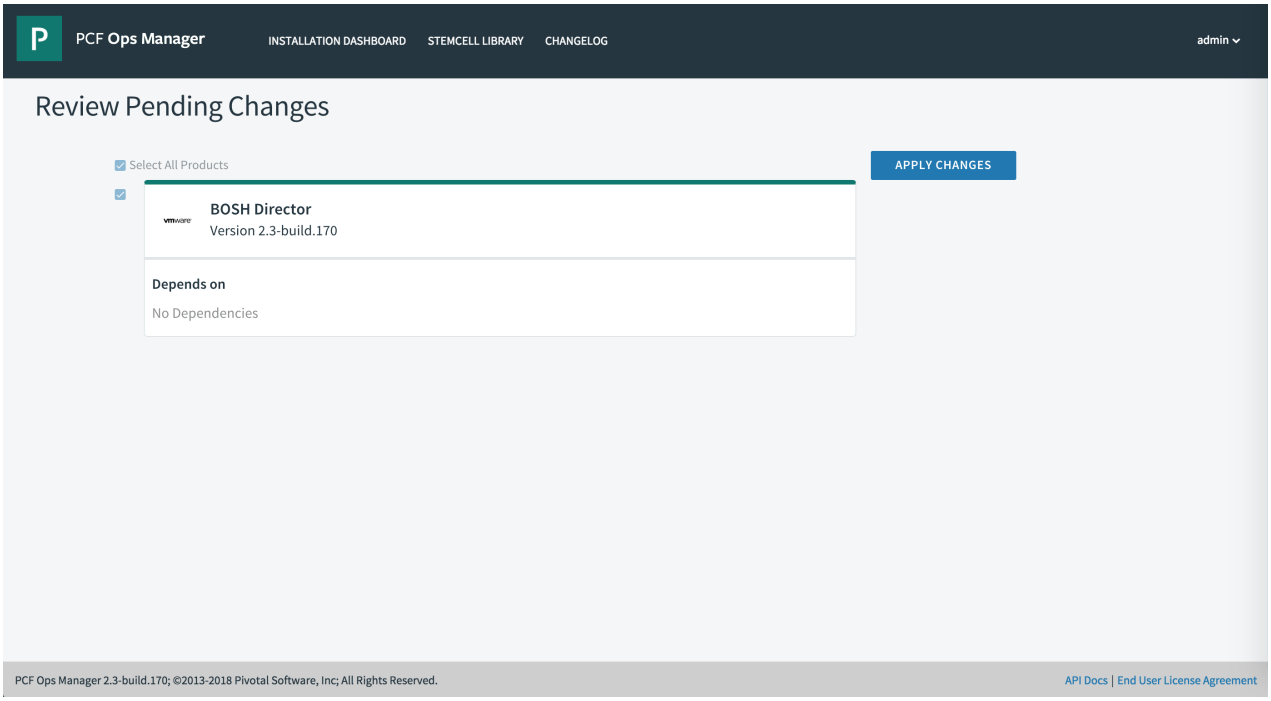
Step 10: Deploy BOSH

Follow the steps below to deploy BOSH:

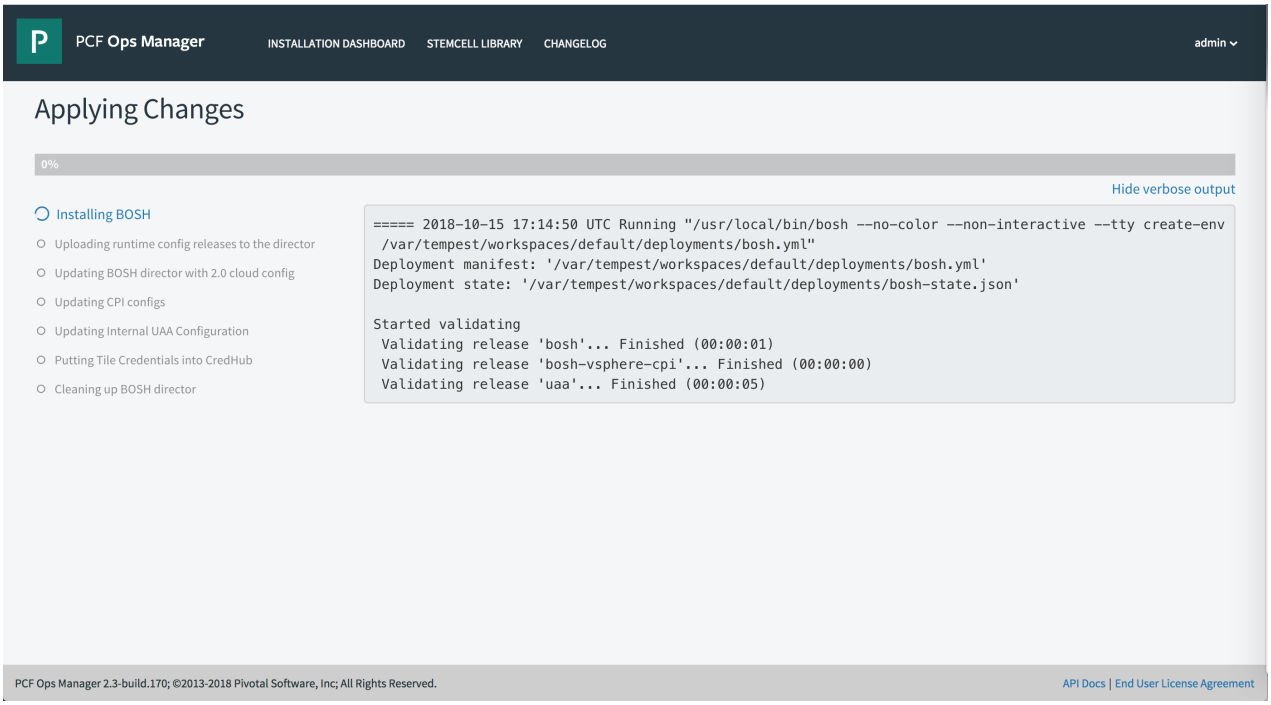
1. Go to the Ops Manager **Installation Dashboard**.



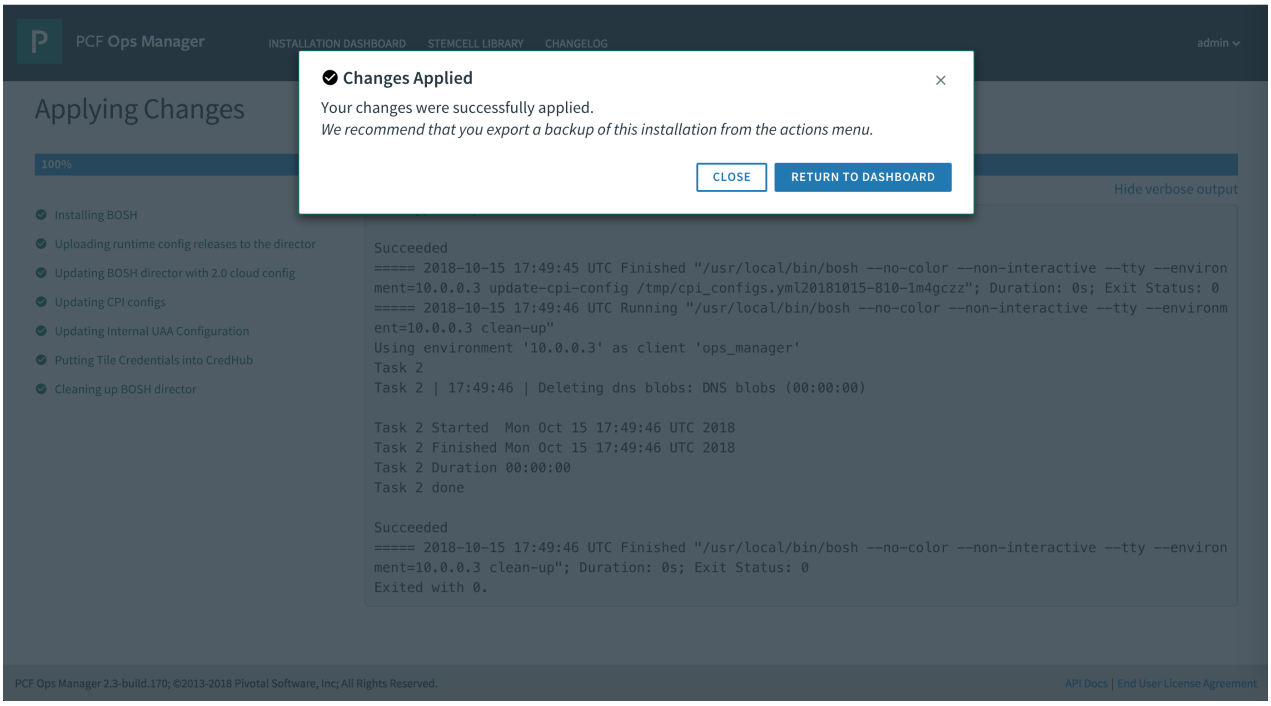
2. Click **Review Pending Changes**.



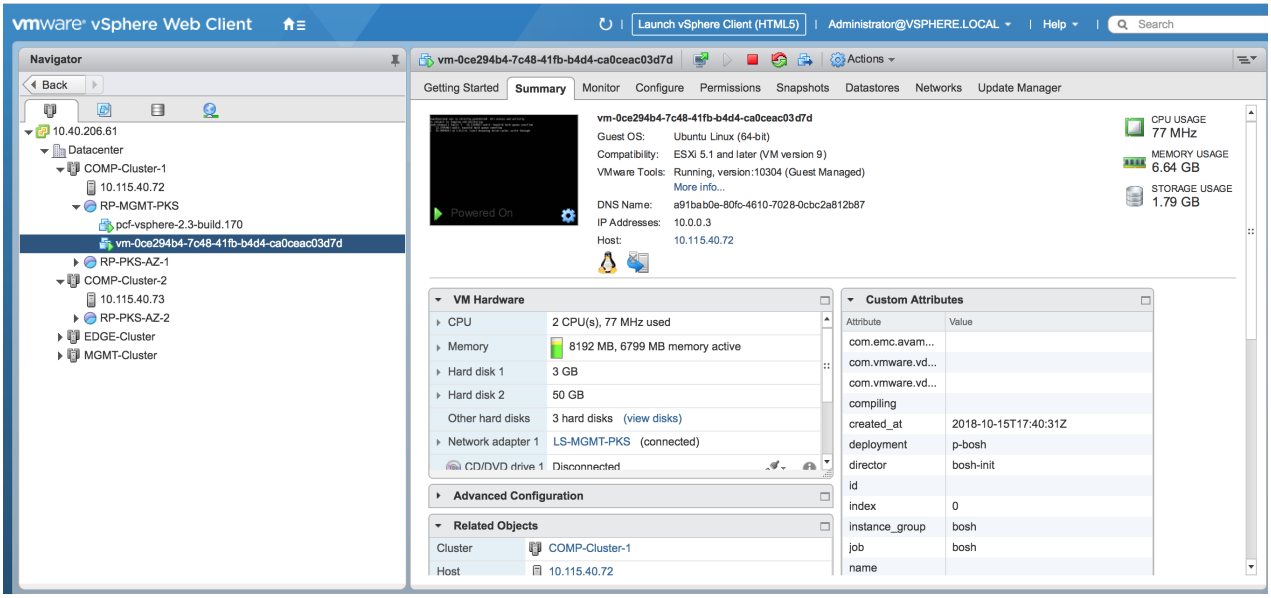
3. Click **Apply Changes**.



4. Confirm changes applied successfully.



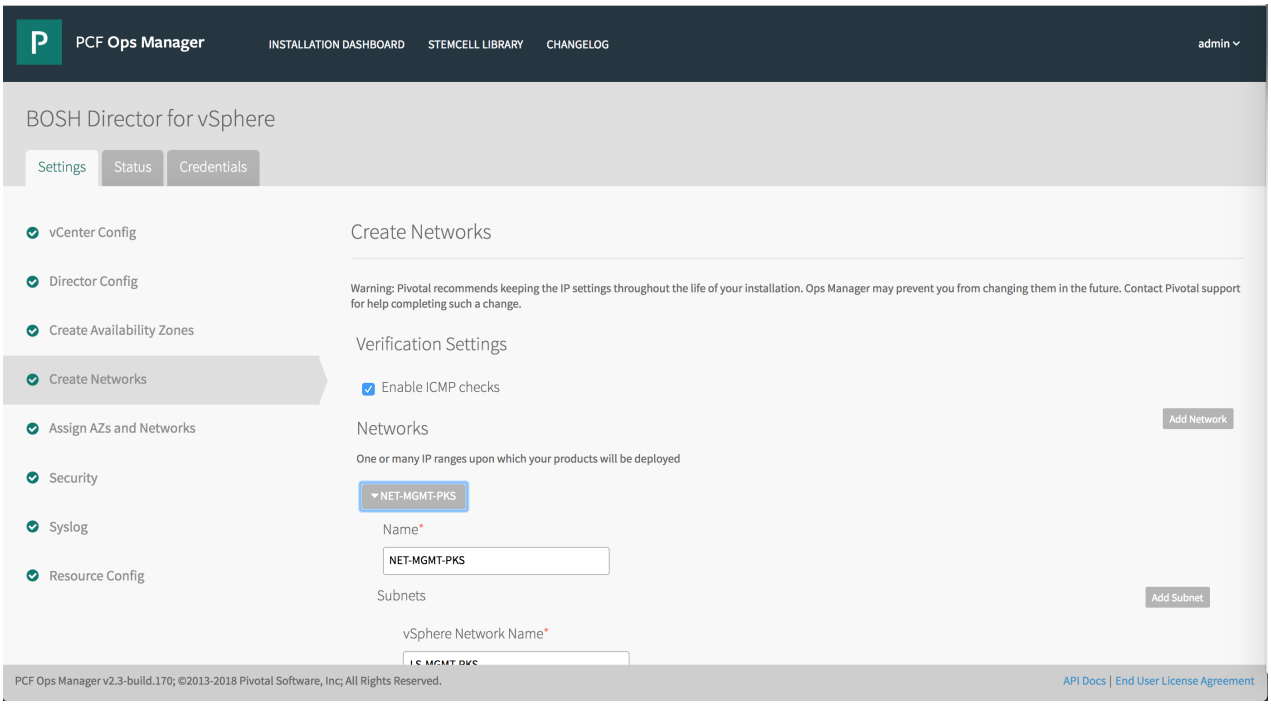
5. Check BOSH VM. Log in to vCenter and check for the `p-bosh` VM deployment in the PKS Management resource pool.



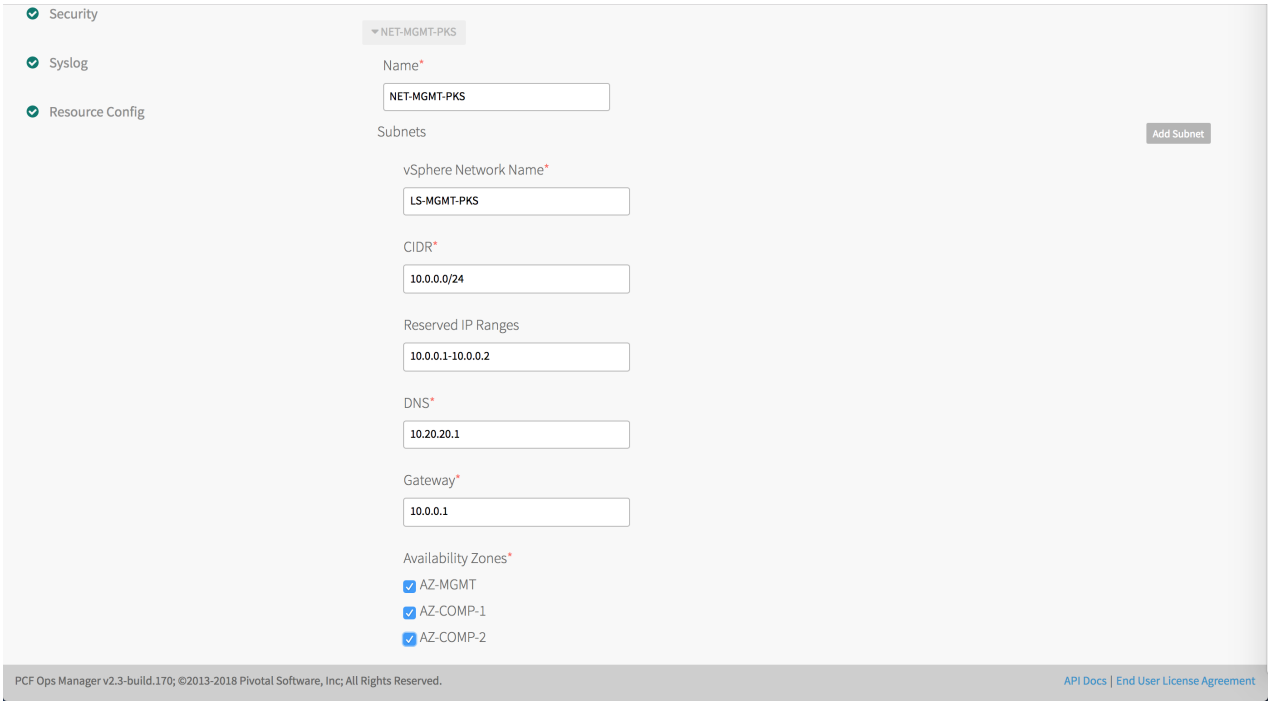
Step 11: Update Network Availability Zones

Ater BOSH is successfully deployed, update the network you defined above (`NET-MGMT-PKS`) to include each of the COMPUTE AZs you defined. This will ensure that both the Management AZ and the Compute AZ(s) appear in the PKS tile for the Plans.

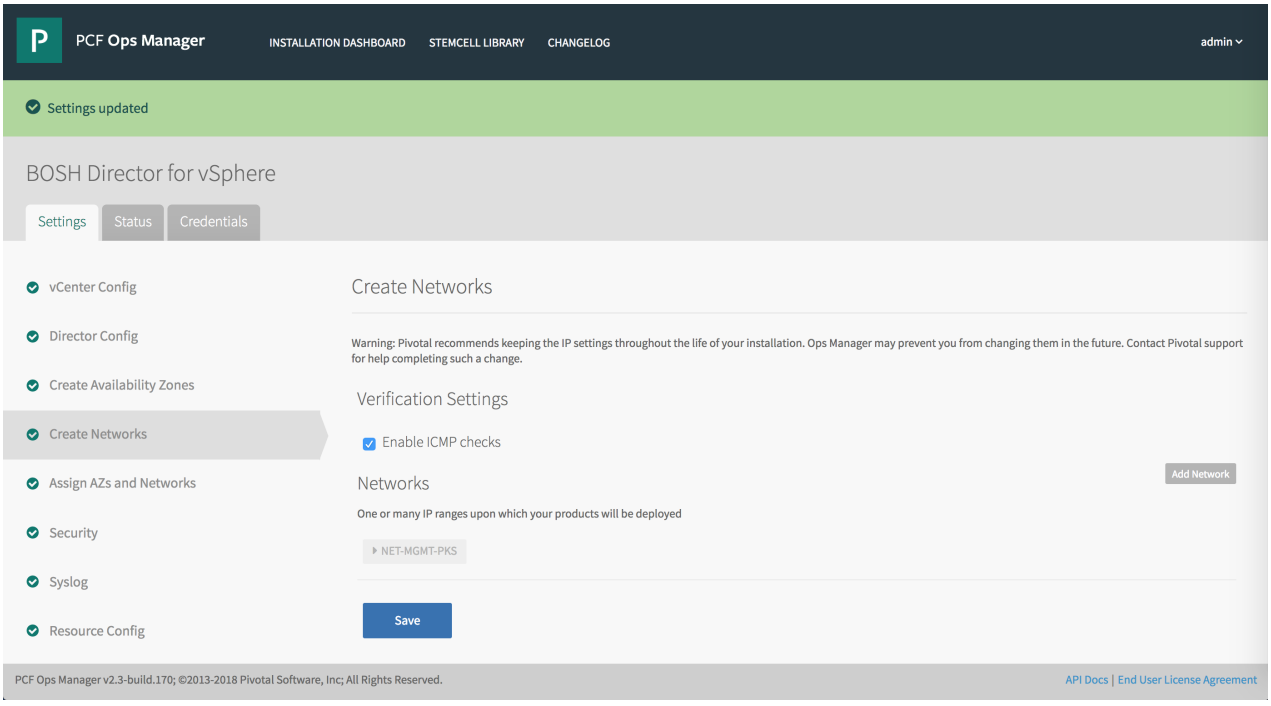
1. Return to the BOSH tile and select **Create Networks**.



2. Edit the network (`NET-MGMT-PKS`) and each COMPUTE AZ.



3. Click Save.



4. Review pending changes and apply them to deploy BOSH.

Next Step

Generate and Register the NSX Manager Superuser Principal Identity Certificate and Key for PKS

Please send any feedback you have to pks-feedback@pivotal.io.

Generating and Registering the NSX Manager Superuser Principal Identity Certificate and Key

Page last updated:

This topic describes how to generate and register the NSX Manager superuser principal identity certificate and key in preparation for installing Pivotal Container Service (PKS) on vSphere with NSX-T.

Prerequisites

Before you begin this procedure, ensure that you have successfully completed all preceding steps for installing PKS on vSphere with NSX-T, including:

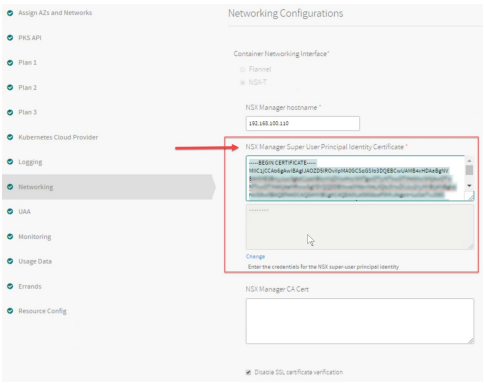
- Deploying NSX-T for PKS
- Creating the PKS Management Plane
- Creating the PKS Compute Plane
- Deploying Ops Manager with NSX-T for PKS
- Generating and Registering the NSX Manager Certificate for PKS
- Configuring BOSH Director with NSX-T for PKS

About the NSX Manager Superuser Principal Identity

The PKS API uses the NSX Manager superuser to communicate with NSX-T to create, delete, and modify networking resources for Kubernetes cluster nodes.

When you configure PKS with NSX-T as the container networking interface, for security purposes you must provide the principal identity certificate and private key for the NSX Manager superuser in the **Networking** pane of the PKS tile.

See the **NSX Manager Super User Principal Identity Certificate** field in the following screenshot:



For more information, see the Networking section of *Installing PKS on vSphere with NSX-T*.

Options for Generating the Certificate and Key

There are two options for generating the principal identity certificate and private key:

- Option A: Use the automatic **Generate RSA Certificate** option in the PKS tile.
- Option B: Run a script on a Linux host with OpenSSL installed that generates the certificate and private key.

Once you have generated the principal identity certificate and key, you must register both with the NSX Manager using an HTTPS POST operation on the NSX API. There is no user interface for this operation.

Option A: Generate and Register the Certificate and Key Using the PKS Tile

Step 1: Generate the Certificate and Key

To generate the certificate and key automatically in the **Networking** pane in the PKS tile, follow the steps below:

1. Navigate to the **Networking** pane in the PKS tile. For more information, see Networking in *Installing PKS on vSphere with NSX-T Integration*.
2. Click **Generate RSA Certificate** and provide a wildcard domain. For example, `*.nsx.pks.vmware.local`.

Step 2: Copy the Certificate and Key to the Linux VM

To copy the certificate and key you generated to a Linux VM, follow the steps below:

Note: The Linux VM must have OpenSSL installed and have network access to the NSX Manager. For example, you can use the PKS client VM where you install the PKS CLI.

1. On the Linux VM you want to use to register the certificate, create a file named `pks-nsx-t-superuser.crt`. Copy the generated certificate into the file.
2. On the Linux VM you want to use to register the key, create a file named `pks-nsx-t-superuser.key`. Copy the generated private key into the file.
3. Save both files.

Step 3: Export Environment Variables

On the Linux VM where you created the certificate and key files, export the environment variables below. Change the `NSX_MANAGER_IP`, `NSX_MANAGER_USERNAME`, and `NSX_MANAGER_PASSWORD` values to match your environment:

```
export NSX_MANAGER="NSX_MANAGER_IP"
export NSX_USER="NSX_MANAGER_USERNAME"
export NSX_PASSWORD='NSX_MANAGER_PASSWORD'
export PI_NAME="pks-nsx-t-superuser"
export NSX_SUPERUSER_CERT_FILE="pks-nsx-t-superuser.crt"
export NSX_SUPERUSER_KEY_FILE="pks-nsx-t-superuser.key"
export NODE_ID=$(cat /proc/sys/kernel/random/uuid)
```

Step 4: Register the Certificate

1. On the same Linux VM, run the following commands to register the certificate with NSX Manager:

```
cert_request=$(cat <<END
  {
    "display_name": "$PI_NAME",
    "pem_encoded": "$(awk '{printf "%s\\n", $0}' $NSX_SUPERUSER_CERT_FILE)"
  }
END
)
```

```
curl -k -X POST \
"https://${NSX_MANAGER}/api/v1/trust-management/certificates?action=import" \
-u "$NSX_USER:$NSX_PASSWORD" \
-H 'content-type: application/json' \
-d "$cert_request"
```

2. Verify that the response includes the `CERTIFICATE_ID` value. You use this value in the following step.

Step 5: Register the Principal Identity

1. On the same Linux VM, export the `CERTIFICATE_ID` environment variable, where the value is the response from the previous step:

```
export CERTIFICATE_ID="CERTIFICATE_ID"
```

2. Register the principal identity with NSX Manager by running the following commands:

```
pi_request=$(cat <<END
  {
    "display_name": "$PI_NAME",
    "name": "$PI_NAME",
    "permission_group": "superusers",
    "certificate_id": "$CERTIFICATE_ID",
    "node_id": "$NODE_ID"
  }
END
)
```

```
curl -k -X POST \
  "https://${NSX_MANAGER}/api/v1/trust-management/principal-identities" \
  -u "$NSX_USER:$NSX_PASSWORD" \
  -H 'content-type: application/json' \
  -d "$pi_request"
```

Step 6: Verify the Certificate and Key

To verify that the certificate and key can be used with NSX-T, run the following command:

```
curl -k -X GET \
"https://${NSX_MANAGER}/api/v1/trust-management/principal-identities" \
--cert $(pwd)/"$NSX_SUPERUSER_CERT_FILE" \
--key $(pwd)/"$NSX_SUPERUSER_KEY_FILE"
```

Option B: Generate and Register the Certificate and Key Using Scripts

This option uses Bash shell scripts to generate and register the NSX Manager superuser principal identity certificate and key.

Note: The Linux VM must have OpenSSL installed and have network access to the NSX Manager. For example, you can use the PKS client VM where you install the PKS CLI.

Step 1: Generate and Register the Certificate and Key

Provided below is the `create_certificate.sh` script that generates a certificate and private key, and then uploads the certificate to the NSX Manager. Complete the following steps to run this script:

1. Log in to a Linux VM in your PKS environment. For example, you can use the PKS client VM.
2. To create an empty file for the first script, run `nano create_certificate.sh`.
3. Copy the following script contents into `create_certificate.sh`, updating the values for the first two lines to match your environment:
    - `NSX_MANAGER_IP`: IP address of the NSX Manager host.
    - `NSX_MANAGER_USERNAME`: Username for NSX Manager.

```
#!/bin/bash
#create_certificate.sh

NSX_MANAGER="NSX_MANAGER_IP"
NSX_USER="NSX_MANAGER_USERNAME"

PI_NAME="pks-nsx-t-superuser"
NSX_SUPERUSER_CERT_FILE="pks-nsx-t-superuser.crt"
NSX_SUPERUSER_KEY_FILE="pks-nsx-t-superuser.key"

stty -echo
printf "Password: "
read NSX_PASSWORD
stty echo

openssl req \
  -newkey rsa:2048 \
  -x509 \
  -nodes \
  -keyout "$NSX_SUPERUSER_KEY_FILE" \
  -new \
  -out "$NSX_SUPERUSER_CERT_FILE" \
  -subj /CN=pks-nsx-t-superuser \
  -extensions client_server_ssl \
  -config <(
    cat /etc/ssl/openssl.cnf \
    <(printf '[client_server_ssl]\nextendedKeyUsage = clientAuth\n')
  ) \
  -sha256 \
  -days 730

cert_request=$(cat <<END
  {
    "display_name": "$PI_NAME",
    "pem_encoded": "$(awk '{printf "%s\\n", $0}' $NSX_SUPERUSER_CERT_FILE)"
  }
END
)

curl -k -X POST \
    "https://${NSX_MANAGER}/api/v1/trust-management/certificates?action=import" \
    -u "$NSX_USER:$NSX_PASSWORD" \
    -H 'content-type: application/json' \
    -d "$cert_request"
```

4. Save the script and run `bash create_certificate.sh`.
5. When prompted, enter the `NSX_MANAGER_PASSWORD` for the NSX user you specified in the script.
6. Complete the following steps to verify the results of the script:
    - The certificate, `pks-nsx-t-superuser.crt`, and private key, `pks-nsx-t-superuser.key`, are generated in the directory where you ran the script.
    - The certificate is uploaded to the NSX Manager and the `CERTIFICATE_ID` value is returned to the console. You need this ID for the second script.

Step 2: Create and Register the Principal Identity

Provided below is the `create_pi.sh` script that creates the principal identity and registers it with the NSX Manager. This script requires the `CERTIFICATE_ID` returned from the `create_certificate.sh` script.

Note: Perform these steps on the same Linux VM where you ran the `create_certificate.sh` script.

1. To create an empty file for the second script, run `nano create_pi.sh`.
2. Copy the following script contents into `create_pi.sh`, updating the values for the first three lines to match your environment:
    - `NSX_MANAGER_IP`: IP address of the NSX Manager host.
    - `NSX_MANAGER_USERNAME`: Username for NSX Manager.
    - `CERTIFICATE_ID`: Response from the `create_certificate.sh` script.

```
#!/bin/bash
#create_pi.sh

NSX_MANAGER="NSX_MANAGER_IP"
NSX_USER="NSX_MANAGER_USERNAME"
CERTIFICATE_ID='CERTIFICATE_ID'

PI_NAME="pks-nsx-t-superuser"
NSX_SUPERUSER_CERT_FILE="pks-nsx-t-superuser.crt"
NSX_SUPERUSER_KEY_FILE="pks-nsx-t-superuser.key"
NODE_ID=$(cat /proc/sys/kernel/random/uuid)

stty -echo
printf "Password: "
read NSX_PASSWORD
stty echo

pi_request=$(cat <<END
   {
      "display_name": "$PI_NAME",
      "name": "$PI_NAME",
      "permission_group": "superusers",
      "certificate_id": "$CERTIFICATE_ID",
      "node_id": "$NODE_ID"
   }
END
)

curl -k -X POST \
   "https://${NSX_MANAGER}/api/v1/trust-management/principal-identities" \
   -u "$NSX_USER:$NSX_PASSWORD" \
   -H 'content-type: application/json' \
   -d "$pi_request"

curl -k -X GET \
   "https://${NSX_MANAGER}/api/v1/trust-management/principal-identities" \
   --cert $(pwd)/"$NSX_SUPERUSER_CERT_FILE" \
   --key $(pwd)/"$NSX_SUPERUSER_KEY_FILE"
```

3. Save the script and run `bash create_pi.sh`.
4. When prompted, enter the `NSX_MANAGER_PASSWORD` for the NSX user you specified in the script.
5. When you configure PKS for deployment, copy and paste the contents of `pks-nsx-t-superuser.crt` and `pks-nsx-t-superuser.key` to the **NSX Manager Super User Principal Identity Certificate** field in the **Networking** pane of the PKS tile. For more information, see the Networking section of *Installing PKS on vSphere with NSX-T*.

Next Step

After you complete this procedure, follow the instructions in Creating NSX-T Objects for PKS.

Please send any feedback you have to pks-feedback@pivotal.io.

Creating NSX-T Objects for PKS

Page last updated:

Installing PKS on vSphere with NSX-T requires the creation of NSX IP blocks for Kubernetes node and pod networks, as well as a Floating IP Pool from which you can assign routable IP addresses to cluster resources.

Create separate NSX-T IP Blocks for the node networks and the pod networks. The subnets for both nodes and pods should have a size of 256 (/16). For more information, see Plan IP Blocks and Reserved IP Blocks.

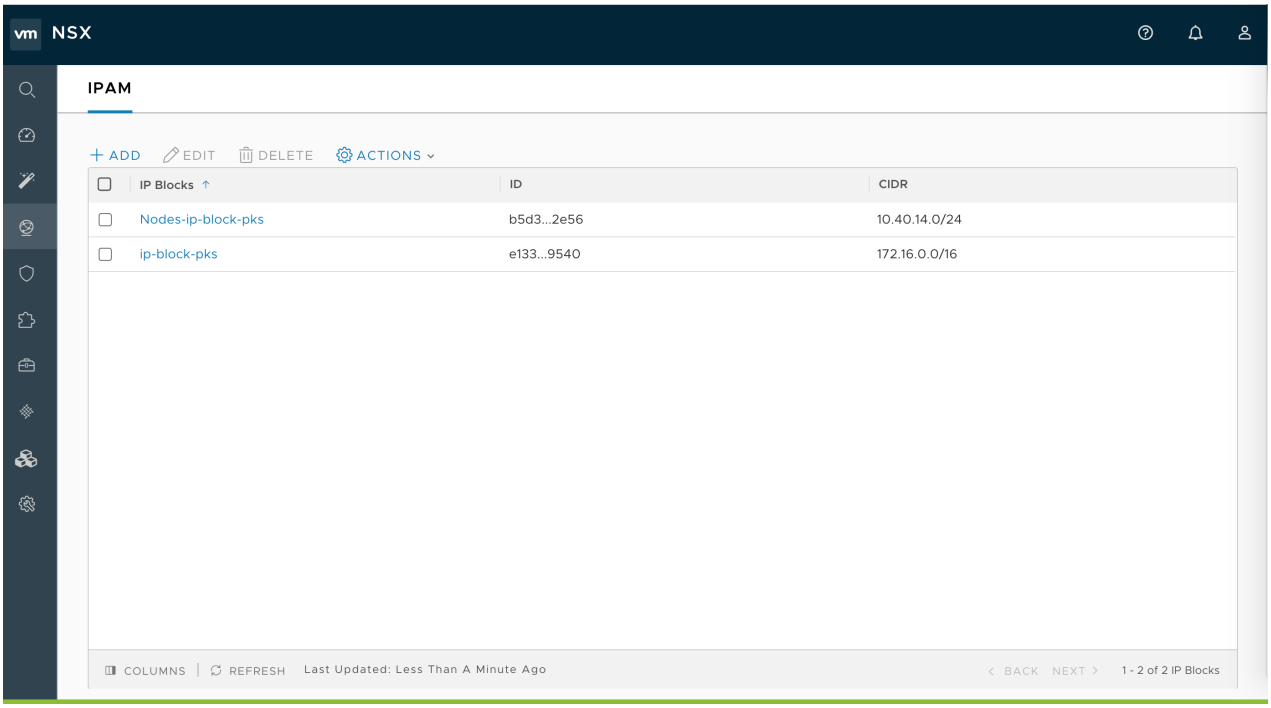
- **NODE-IP-BLOCK** is used by PKS to assign address space to Kubernetes master and worker nodes when new clusters are deployed or a cluster increases its scale.
- **POD-IP-BLOCK** is used by the NSX-T Container Plug-in (NCP) to assign address space to Kubernetes pods through the Container Networking Interface (CNI).

In addition, create a Floating IP Pool from which to assign routable IP addresses to components. This network provides your load balancing address space for each Kubernetes cluster created by PKS. The network also provides IP addresses for Kubernetes API access and Kubernetes exposed services. For example, `10.172.2.0/24` provides 256 usable IPs. This network is used when creating the virtual IP pools, or when the services are deployed. You enter this network in the **Floating IP Pool ID** field in the **Networking** pane of the PKS tile.

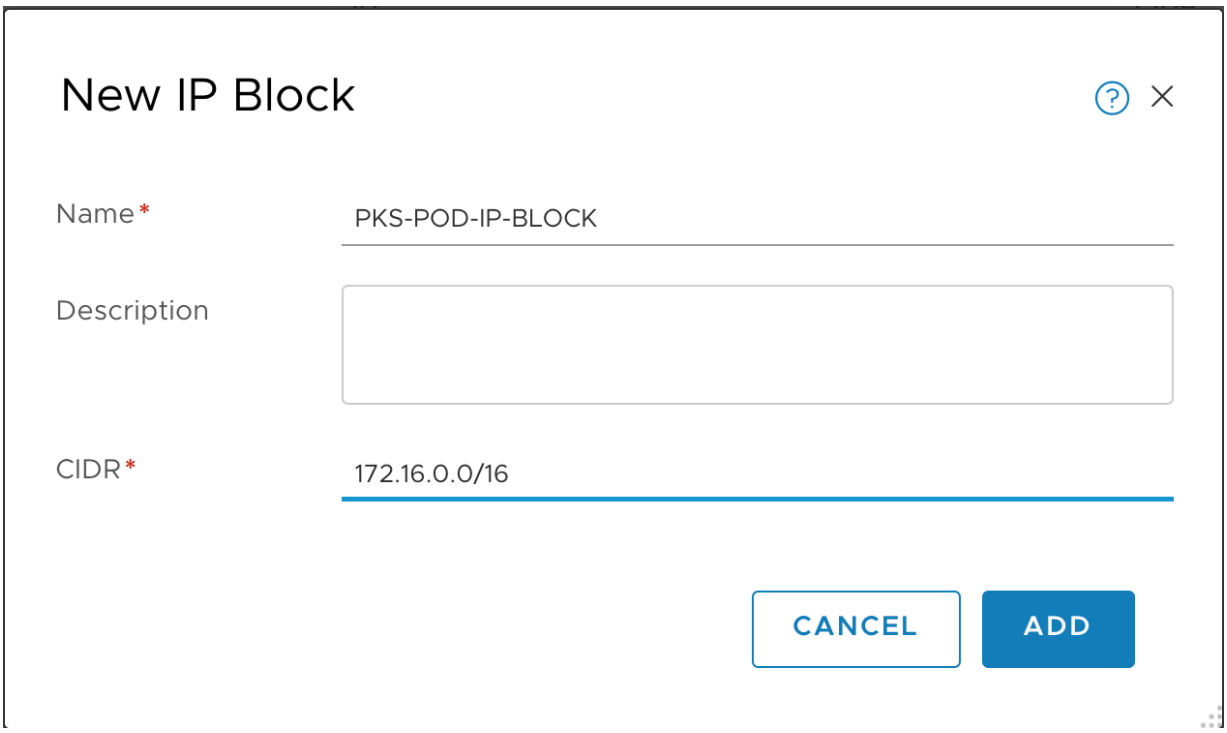
Complete the following instructions to create the required NSX-T network objects.

Create the Pods IP Block

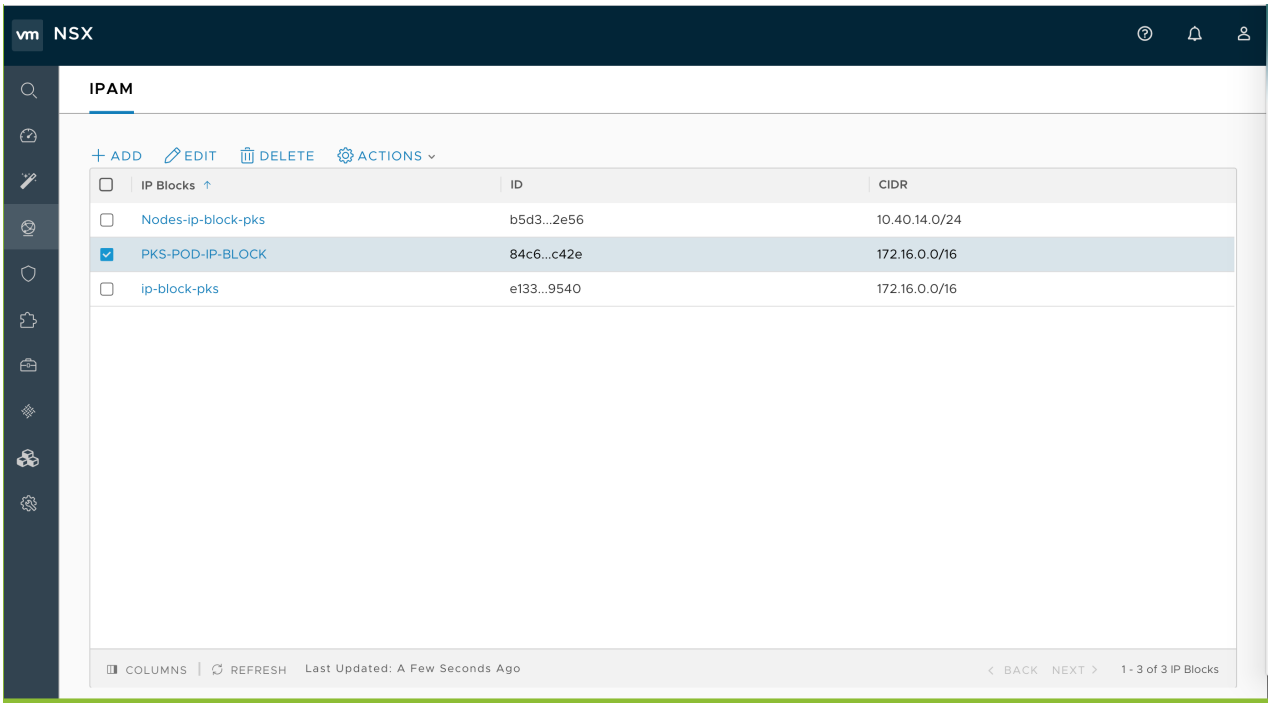
1. In NSX Manager, go to **Networking > IPAM**.



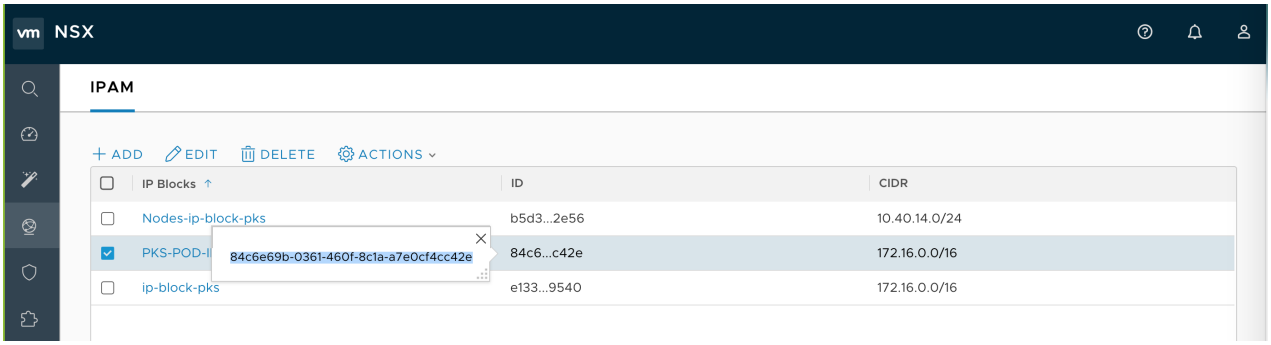
2. Add a new IP Block for Pods. For example:
    - **Name**: PKS-PODS-IP-BLOCK
    - **CIDR**: 172.16.0.0/16



3. Verify creation of the Pods IP Block.

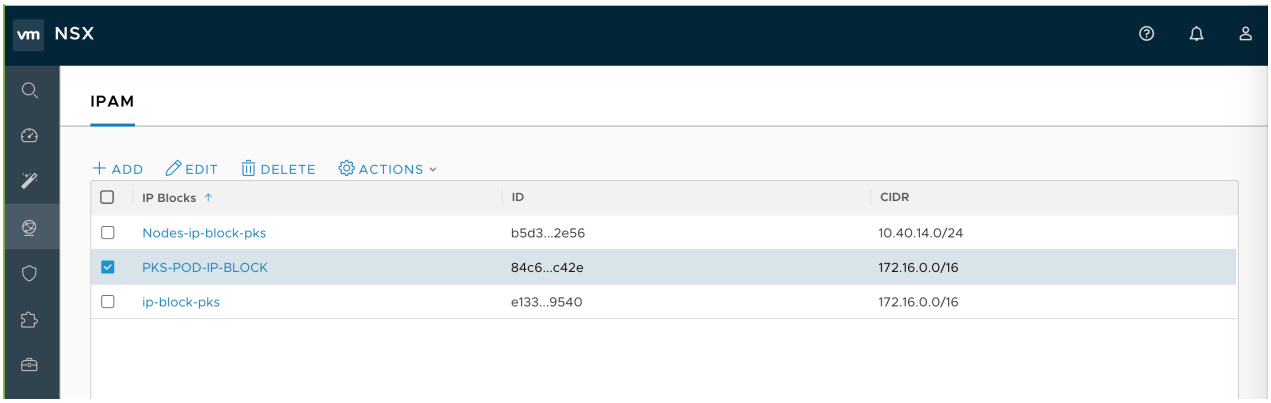


4. Get the UUID of the Pods IP Block object. You use this UUID when you install PKS with NSX-T.



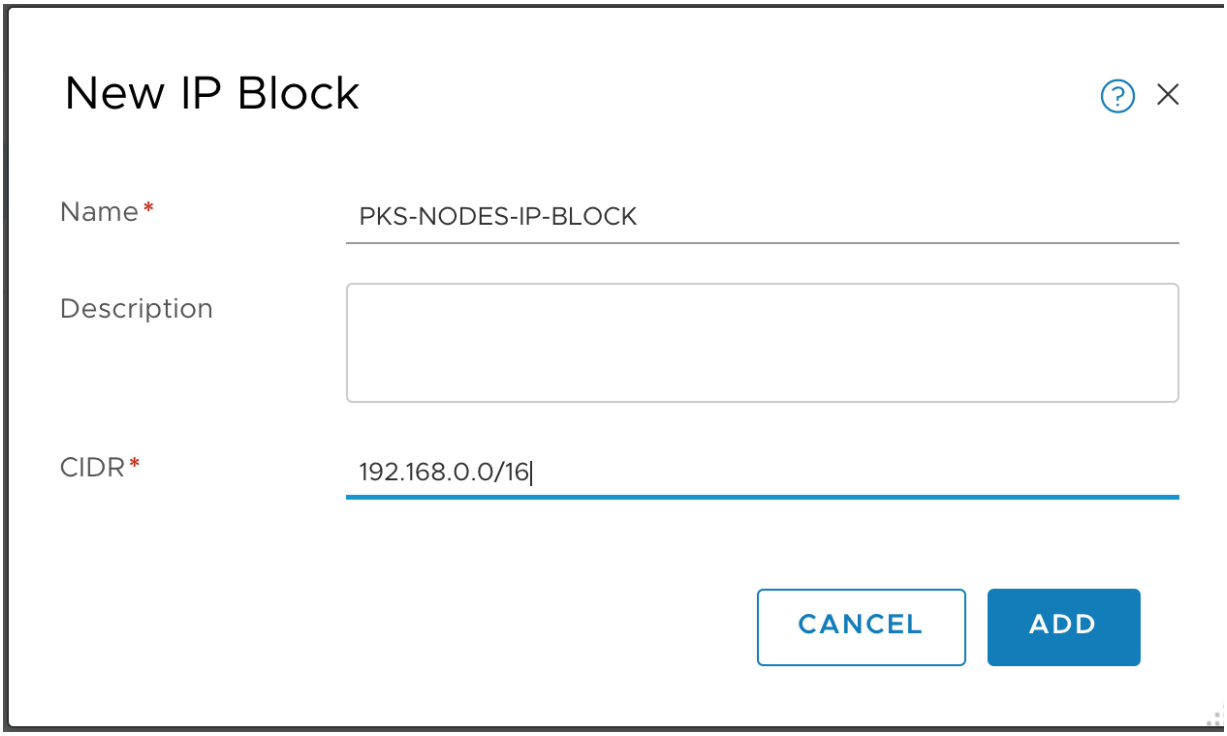
Create the Nodes IP Block

1. In NSX Manager, go to **Networking > IPAM**.

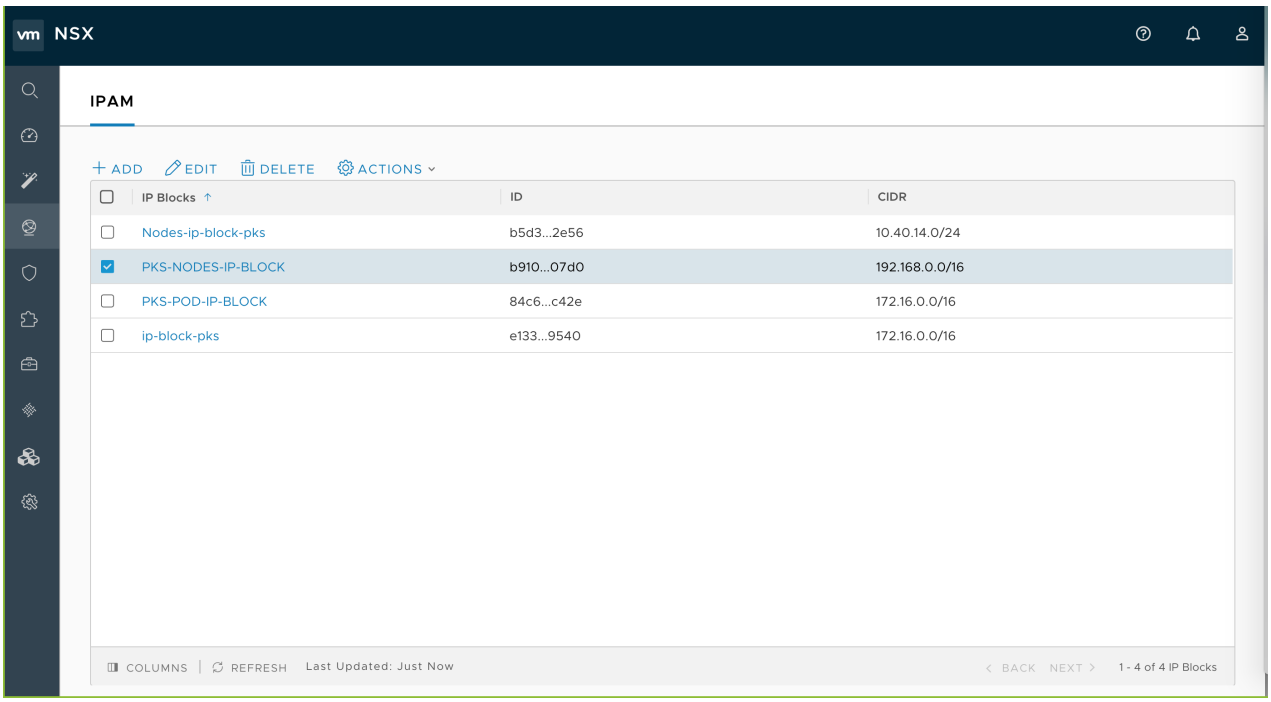


2. Add a new IP Block for Nodes. For example:

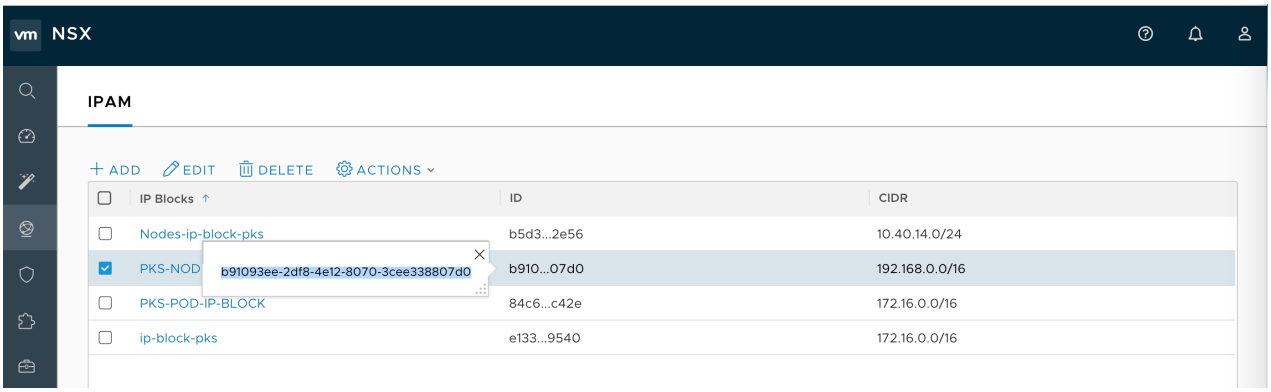
    - **Name**: PKS-NODES-IP-BLOCK
    - **CIDR**: 192.168.0.0/16



3. Verify creation of the Nodes IP Block.

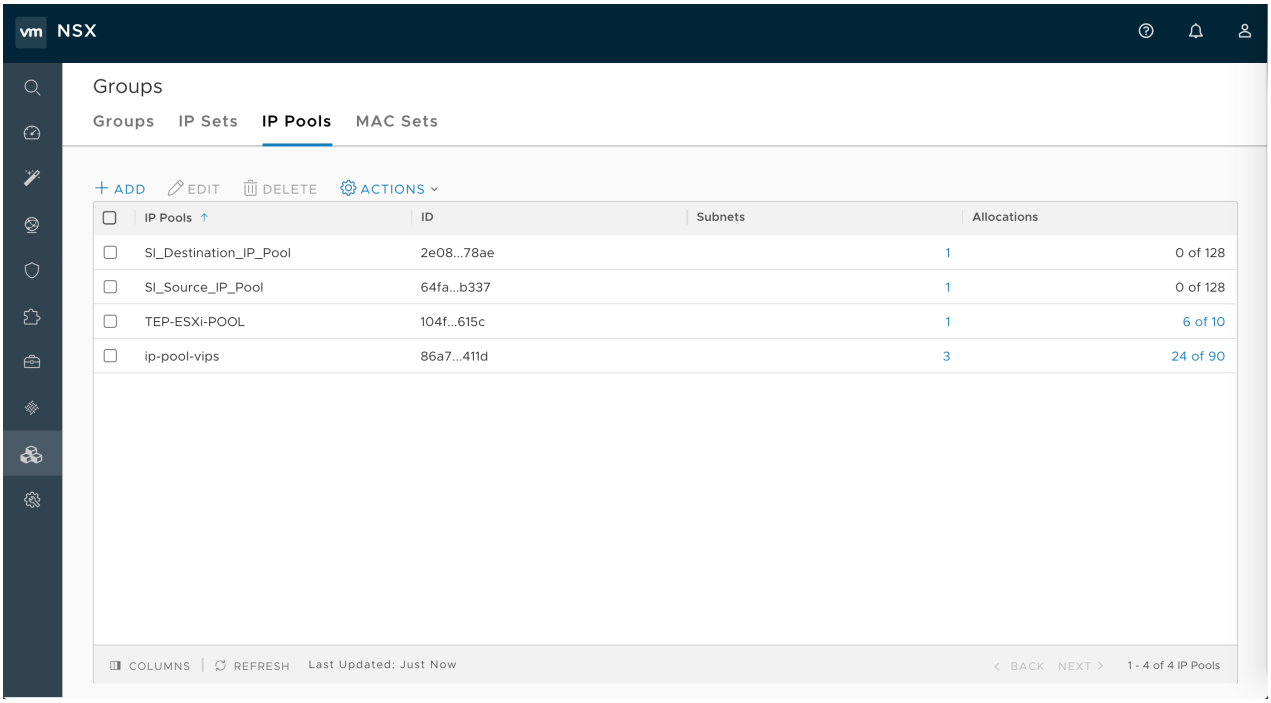


4. Get the UUID of the Nodes IP Block object. You use this UUID when you install PKS with NSX-T.



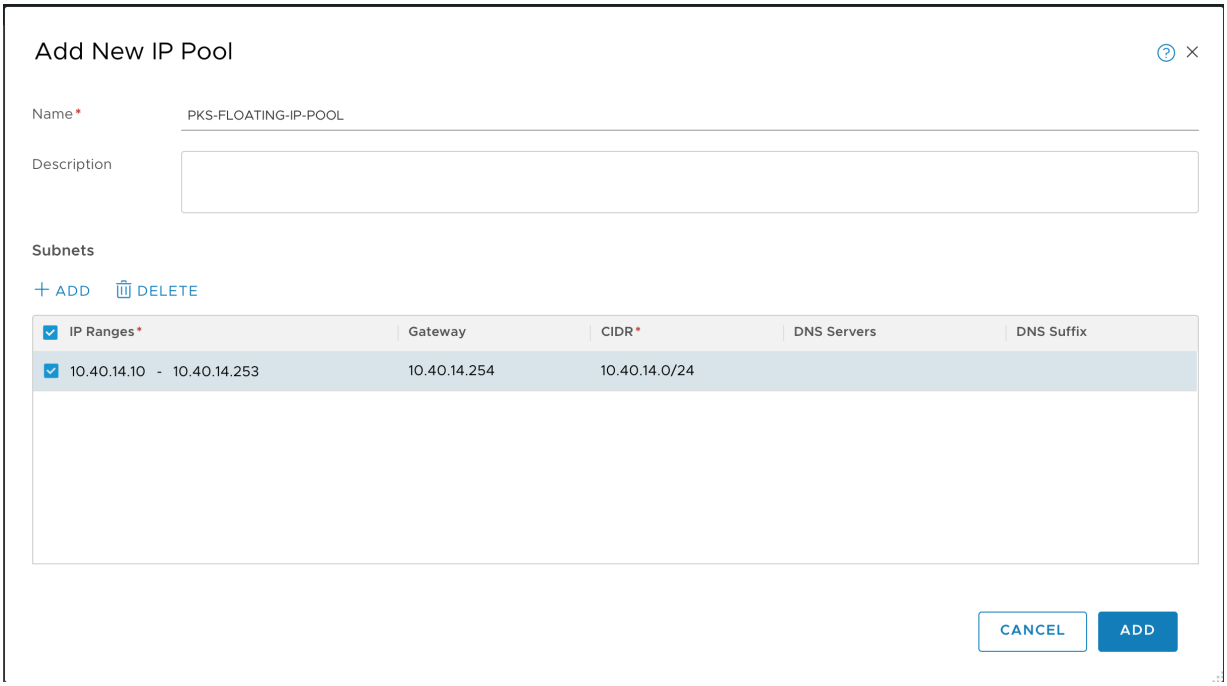
Create Floating IP Pool

1. In NSX Manager, go to **Inventory > Groups > IP Pool**.

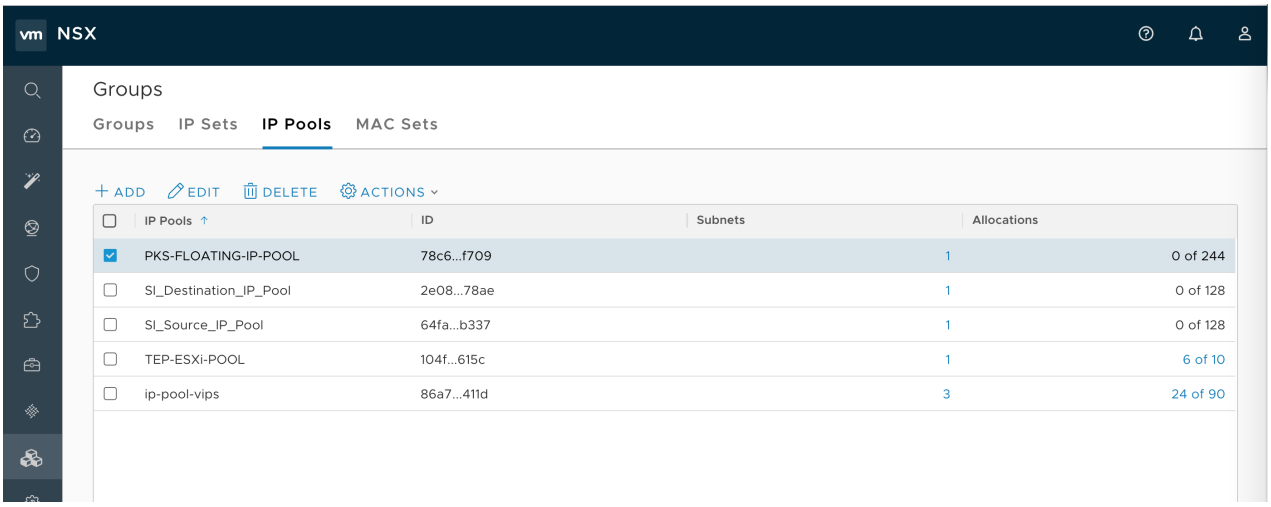


2. Add a new Floating IP Pool. For example:

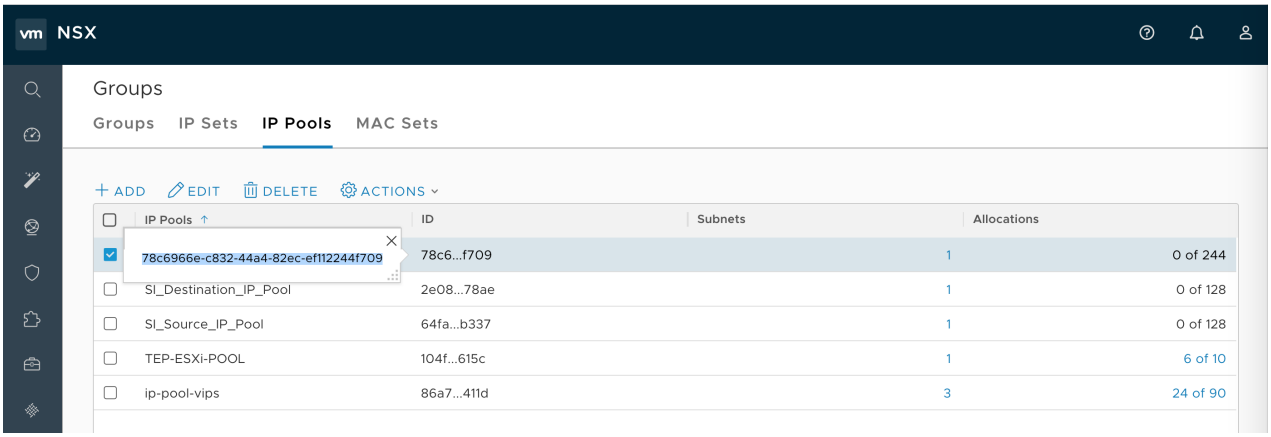
   - **Name**: PKS-FLOATING-IP-POOL
   - **IP Ranges**: 10.40.14.10 - 10.40.14.253
   - **Gateway**: 10.40.14.254
   - **CIDR**: 10.40.14.0/24



3. Verify creation of the Nodes IP Block.



4. Get the UUID of the Floating IP Pool object. You use this UUID when you install PKS with NSX-T.



Next Step

After you complete this procedure, follow the instructions in Installing PKS on vSphere with NSX-T.

Please send any feedback you have to pks-feedback@pivotal.io.

Installing PKS on vSphere with NSX-T

Page last updated:

This topic describes how to install and configure Pivotal Container Service (PKS) on vSphere with NSX-T integration.

Prerequisites

Before you begin this procedure, ensure that you have successfully completed all preceding steps for installing PKS on vSphere with NSX-T, including:

- Deploying NSX-T for PKS
- Creating the PKS Management Plane
- Creating the PKS Compute Plane
- Deploying Ops Manager with NSX-T for PKS
- Generating and Registering the NSX Manager Certificate for PKS
- Configuring BOSH Director with NSX-T for PKS
- Generating and Registering the NSX Manager Superuser Principal Identity Certificate and Key for PKS
- Creating NSX-T Objects for PKS

Step 1: Install PKS

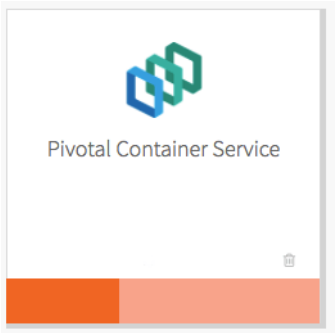
To install PKS, do the following:

1. Download the product file from Pivotal Network.
2. Navigate to `https://YOUR-OPS-MANAGER-FQDN/` in a browser to log in to the Ops Manager Installation Dashboard.
3. Click **Import a Product** to upload the product file.
4. Under **Pivotal Container Service** in the left column, click the plus sign to add this product to your staging area.

Step 2: Configure PKS

Click the orange **Pivotal Container Service** tile to start the configuration process.

Note: Configuration of NSX-T or Flannel **cannot** be changed after initial installation and configuration of PKS.



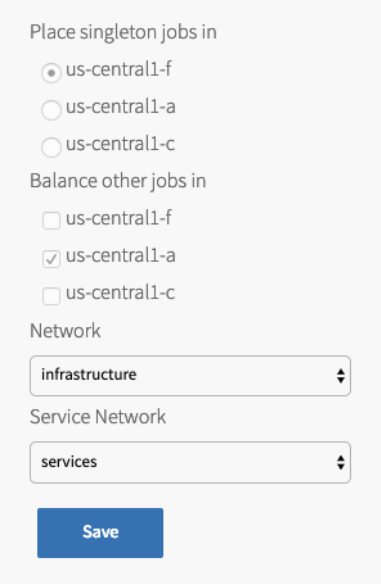
WARNING: When you configure the PKS tile, do not use spaces in any field entries. This includes spaces between characters as well as leading and trailing spaces. If you use a space in any field entry, the deployment of PKS fails.

Assign AZs and Networks

Perform the following steps:

1. Click **Assign AZs and Networks**.
2. Select the availability zone (AZ) where you want to deploy the PKS API VM as a singleton job.

> **Note**: You must select an additional AZ for balancing other jobs before clicking **Save**, but this selection has no effect in the current version of PKS.

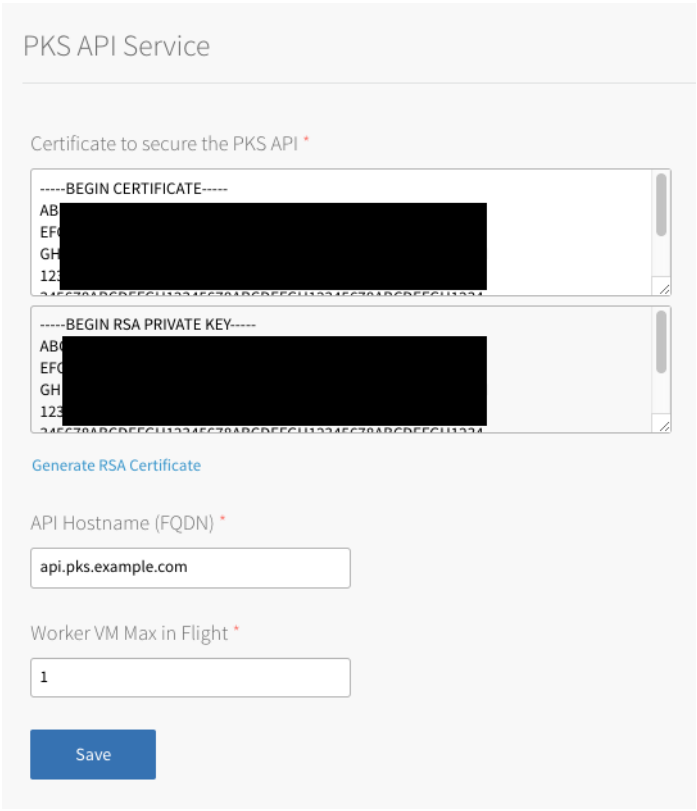


3. Under **Network**, select the PKS Management Network linked to the `ls-pks-mgmt` NSX-T logical switch you created in the Create Networks Page step of *Configuring BOSH Director with NSX-T for PKS*. This will provide network placement for the PKS API VM.
4. Under **Service Network**, your selection depends on whether you are installing a new PKS deployment or upgrading from a previous version of PKS.
    - If you are deploying PKS with NSX-T for the first time, select the PKS Managment Network you specified in the **Network** field. You do not need to create or define a service network because PKS creates the service network for you during the installation process.
    - If you are upgrading from a previous version of PKS, then select the **Service Network** linked to the `ls-pks-service` NSX-T logical switch that PKS created for you during installation. The service network provides network placement for existing on-demand Kubernetes cluster service instances that were created by the PKS broker.
5. Click **Save**.

PKS API

Perform the following steps:

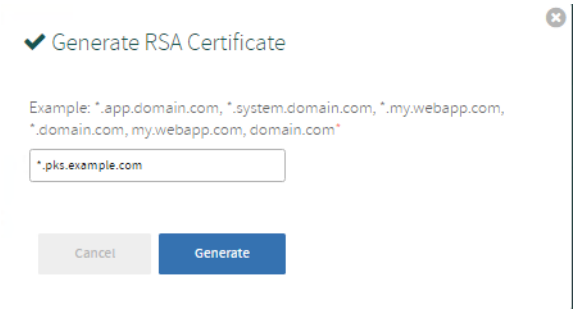
1. Click **PKS API**.
2. Under **Certificate to secure the PKS API**, provide your own certificate and private key pair.



The certificate that you supply should cover the domain that routes to the PKS API VM with TLS termination on the ingress.

If you do not have a certificate and private key pair, PKS can generate one for you. To generate a certificate, do the following:

a. Select the **Generate RSA Certificate** link.
b. Enter the domain for your API hostname. This can be a standard FQDN or a wildcard domain.
c. Click **Generate**.



3. Under **API Hostname (FQDN)**, enter the FQDN that you registered to point to the PKS API load balancer, such as `api.pks.example.com`. To retrieve the public IP address or FQDN of the PKS API load balancer, log in to your IaaS console.

4. Under **Worker VM Max in Flight**, enter the maximum number of non-canary worker instances to create or resize in parallel within an availability zone.

   This field sets the `max_in_flight` variable, which limits how many instances of a component can start simultaneously when a cluster is created or resized. The variable defaults to `1`, which means that only one component starts at a time.

5. Click **Save**.

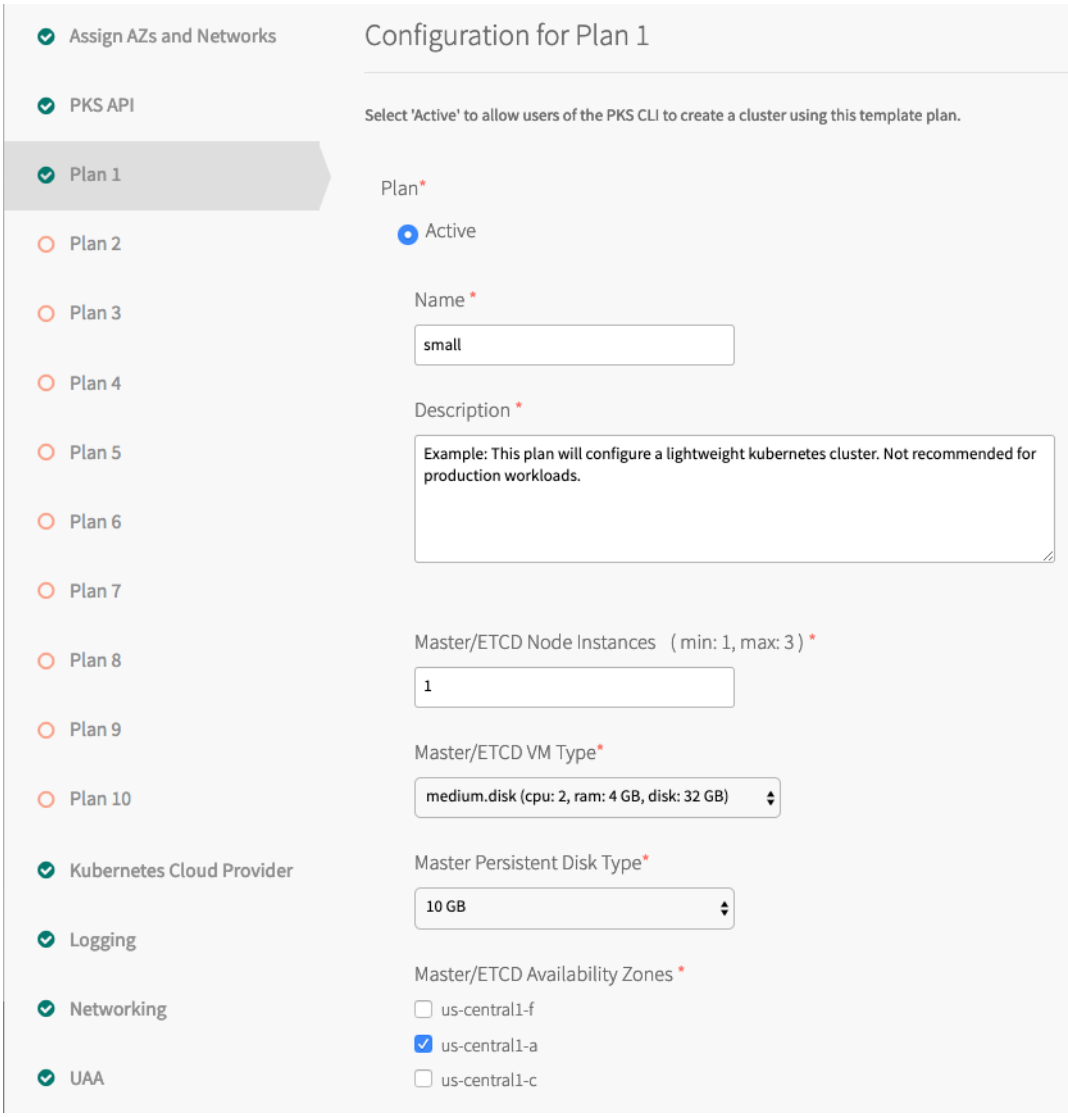
Plans

To activate a plan, perform the following steps:

1. Click the plan that you want to activate.

> **Note**: A plan defines a set of resource types used for deploying clusters. You can configure up to ten plans. You must configure **Plan 1**.

2. Select **Active** to activate the plan and make it available to developers deploying clusters.



3. Under **Name**, provide a unique name for the plan.

4. Under **Description**, edit the description as needed. The plan description appears in the Services Marketplace, which developers can access by using PKS CLI.

5. Under **Master/ETCD Node Instances**, select the default number of Kubernetes master/etcd nodes to provision for each cluster. You can enter either `1` or `3`.

> **Note**: If you deploy a cluster with multiple master/etcd node VMs, confirm that you have sufficient hardware to handle the increased load on disk write and network traffic. For more information, see Hardware recommendations in the etcd documentation.
>
> In addition to meeting the hardware requirements for a multi-master cluster, we recommend configuring monitoring for etcd to monitor disk latency, network latency, and other indicators for the health of the cluster. For more information, see Monitoring Master/etcd Node VMs.

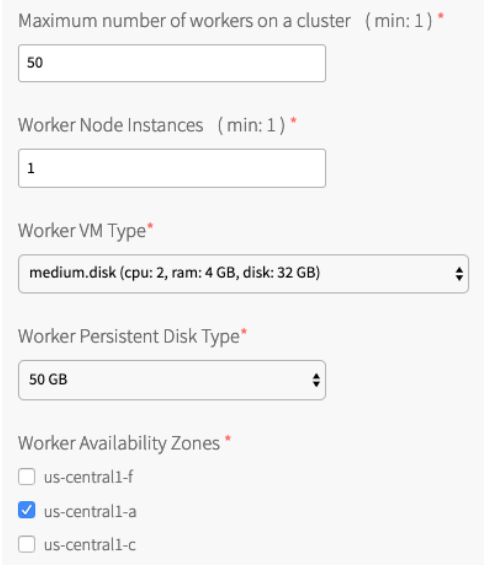
> **WARNING**: To change the number of master/etcd nodes for a plan, you must ensure that no existing clusters use the plan. PKS does not support changing the number of master/etcd nodes for plans with existing clusters.

6. Under **Master/ETCD VM Type**, select the type of VM to use for Kubernetes master/etcd nodes. For more information, including master node VM customization options, see the Master Node VM Size section of *VM Sizing for PKS Clusters*.

7. Under **Master Persistent Disk Type**, select the size of the persistent disk for the Kubernetes master node VM.

8. Under **Master/ETCD Availability Zones**, select one or more AZs for the Kubernetes clusters deployed by PKS. If you select more than one AZ, PKS deploys the master VM in the first AZ and the worker VMs across the remaining AZs.

9. Under **Maximum number of workers on a cluster**, set the maximum number of Kubernetes worker node VMs that PKS can deploy for each cluster. Enter any whole number in this field.



10. Under **Worker Node Instances**, select the default number of Kubernetes worker nodes to provision for each cluster.

    If the user creating a cluster with the PKS CLI does not specify a number of worker nodes, the cluster is deployed with the default number set in this field. This value cannot be greater than the maximum worker node value you set in the previous field. For more information about creating clusters, see Creating Clusters.

    For high availability, create clusters with a minimum of three worker nodes, or two per AZ if you intend to use PersistentVolumes (PVs). For example, if you deploy across three AZs, you should have six worker nodes. For more information about PVs, see PersistentVolumes in *Maintaining Workload Uptime*. Provisioning a minimum of three worker nodes, or two nodes per AZ is also recommended for stateless workloads.

    If you later reconfigure the plan to adjust the default number of worker nodes, the existing clusters that have been created from that plan are not automatically upgraded with the new default number of worker nodes.

11. Under **Worker VM Type**, select the type of VM to use for Kubernetes worker node VMs. For more information, including worker node VM customization options, see the Worker Node VM Number and Size section of *VM Sizing for PKS Clusters*.

    Note: If you install PKS in an NSX-T environment, we recommend that you select a **Worker VM Type** with a minimum disk size of 16 GB. The disk space provided by the default `medium` Worker VM Type is insufficient for PKS with NSX-T.

12. Under **Worker Persistent Disk Type**, select the size of the persistent disk for the Kubernetes worker node VMs.

13. Under **Worker Availability Zones**, select one or more AZs for the Kubernetes worker nodes. PKS deploys worker nodes equally across the AZs you select.

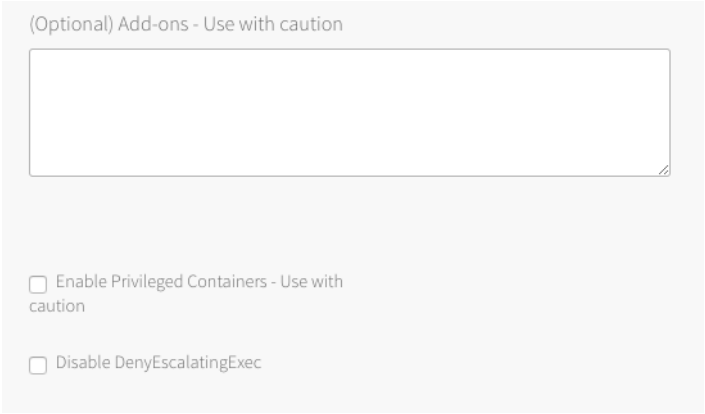
14. Under **Kubelet customization - system-reserved**, enter resource values that Kubelet can use to reserve resources for system daemons. For example, `memory=250Mi, cpu=150m`. For more information about system-reserved values, see the Kubernetes documentation.

15. Under **Kubelet customization - eviction-hard**, enter threshold limits that Kubelet can use to evict pods when they exceed the limit. Enter limits in the format `EVICTION-SIGNAL=QUANTITY`. For example, `memory.available=100Mi, nodefs.available=10%, nodefs.inodesFree=5%`. For more information about eviction thresholds, see the Kubernetes documentation.

    WARNING: Use the Kubelet customization fields with caution. If you enter values that are invalid or that exceed the limits the system supports, Kubelet might fail to start. If Kubelet fails to start, you cannot create clusters.

16. Under **Errand VM Type**, select the size of the VM that contains the errand. The smallest instance possible is sufficient, as the only errand running on this VM is the one that applies the **Default Cluster App** YAML configuration.

17. (Optional) Under **(Optional) Add-ons - Use with caution**, enter additional YAML configuration to add custom workloads to each cluster in this plan. You can specify multiple files using `---` as a separator. For more information, see Adding Custom Workloads.



18. (Optional) To allow users to create pods with privileged containers, select the **Enable Privileged Containers - Use with caution** option. For more information, see Pods in the Kubernetes documentation.
19. (Optional) To disable the admission controller, select the **Disable DenyEscalatingExec** checkbox. If you select this option, clusters in this plan can create security vulnerabilities that may impact other tiles. Use this feature with caution.
20. Click **Save**.

To deactivate a plan, perform the following steps:

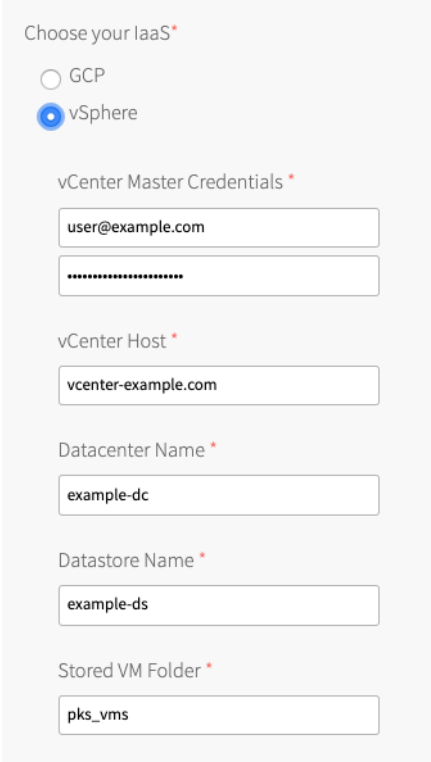
1. Click the plan that you want to deactivate.
2. Select **Inactive**.
3. Click **Save**.

Kubernetes Cloud Provider

In the procedure below, you use credentials for vCenter master VMs. You must have provisioned the service account with the correct permissions. For more information, see Create the Master Node Service Account in *Preparing vSphere Before Deploying PKS*.

To configure your Kubernetes cloud provider settings, follow the procedure below:

1. Click **Kubernetes Cloud Provider**.
2. Under **Choose your IaaS**, select **vSphere**.
3. Ensure the values in the following procedure match those in the **vCenter Config** section of the Ops Manager tile.



a. Enter your **vCenter Master Credentials**. Enter the username using the format `user@example.com`. For more information about the master node service account, see Preparing to Deploy PKS on vSphere.
b. Enter your **vCenter Host**. For example, `vcenter-example.com`.
c. Enter your **Datacenter Name**. For example, `example-dc`.
d. Enter your **Datastore Name**. For example, `example-ds`.
e. Enter the **Stored VM Folder** so that the persistent stores know where to find the VMs. To retrieve the name of the folder, navigate to your BOSH Director tile, click **vCenter Config**, and locate the value for **VM Folder**. The default folder name is `pcf_vms`.
f. Click **Save**.

Note: The value for the **Datastore Name** field is intended to be a single datastore that is the default target. This field should not include a list of BOSH Job/VMDK datastores. The default datastore is used if the Kubernetes cluster `StorageClass` does not define a `StoragePolicy`. For more information, see PersistentVolume Storage Options on vSphere.

Note: For multi-AZ and multi-cluster environments, we recommend using a shared datastore that is available to each vSphere cluster, as opposed to a datastore that is local to a single cluster. For more information, see PersistentVolume Storage Options on vSphere.

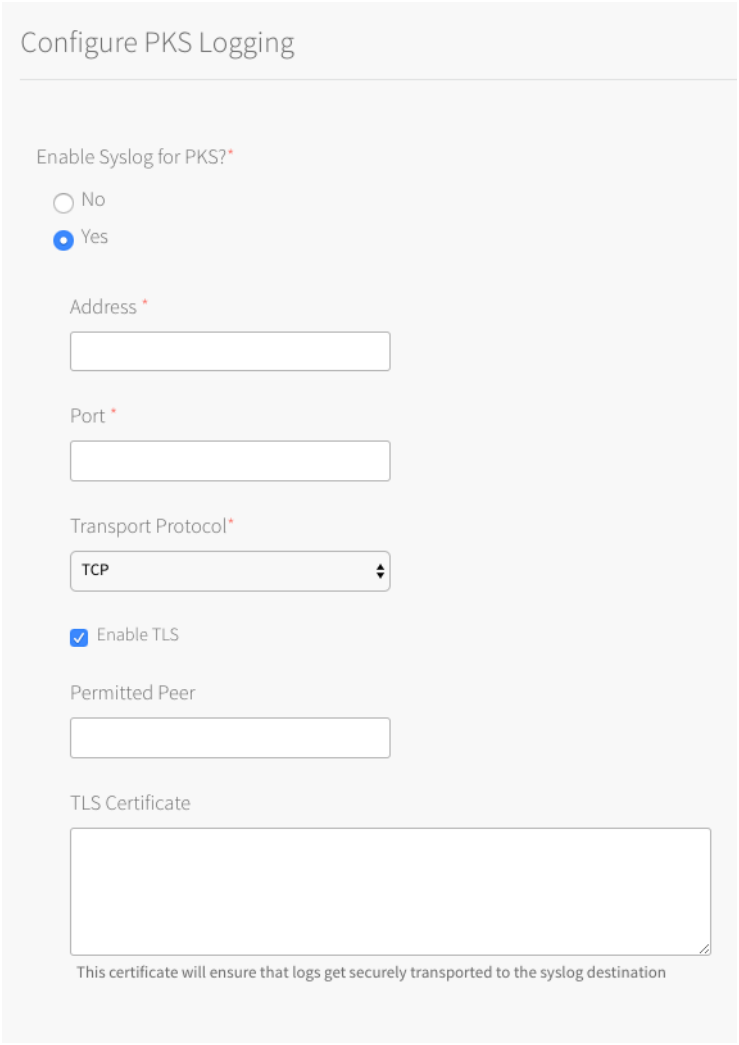
(Optional) Logging

You can designate an external syslog endpoint for forwarded BOSH-deployed VM logs.

In addition, you can enable sink resources to collect PKS cluster and namespace log messages.

To configure logging in PKS, do the following:

1. Click **Logging**.
2. To enable syslog forwarding for BOSH-deployed VM logs, select **Yes**.



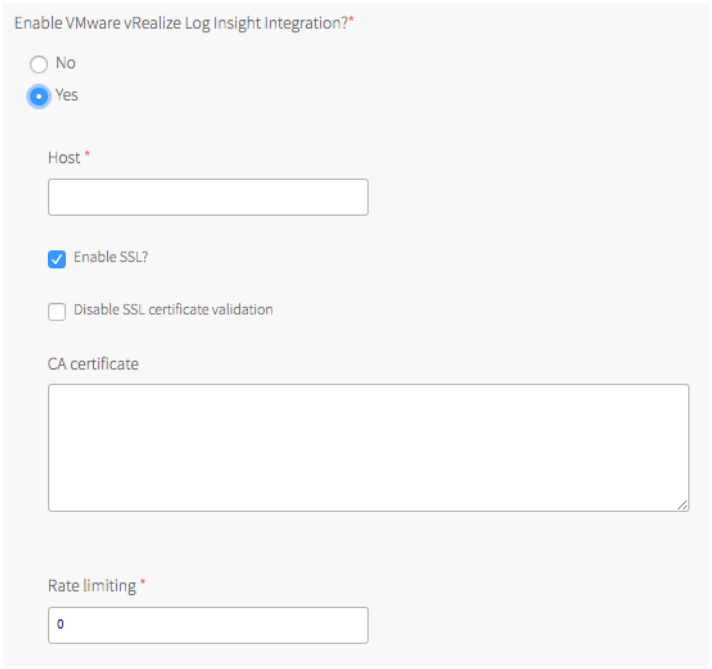
3. Under **Address**, enter the destination syslog endpoint.
4. Under **Port**, enter the destination syslog port.
5. Select a transport protocol for log forwarding.
6. (Optional) Pivotal strongly recommends that you enable TLS encryption when forwarding logs as they may contain sensitive information. For example, these logs may contain cloud provider credentials. To enable TLS, perform the following steps:
    a. Under **Permitter Peer**, provide the accepted fingerprint (SHA1) or name of remote peer. For example, `*.YOUR-LOGGING-SYSTEM.com`.
    b. Under **TLS Certificate**, provide a TLS certificate for the destination syslog endpoint.

    > **Note**: You do not need to provide a new certificate if the TLS certificate for the destination syslog endpoint is signed by a Certificate Authority (CA) in your BOSH certificate store.

7. You can manage logs using VMware vRealize Log Insight (vRLI). The integration pulls logs from all BOSH jobs and containers running in the cluster, including node logs from core Kubernetes and BOSH processes, Kubernetes event logs, and POD stdout and stderr.

> **Note**: Before you configure the vRLI integration, you must have a vRLI license and vRLI must be installed, running, and available in your environment. You need to provide the live instance address during configuration. For instructions and additional information, see the vRealize Log Insight documentation.

By default, vRLI logging is disabled. To enable and configure vRLI logging, under **Enable VMware vRealize Log Insight Integration?**, select **Yes** and



then perform the following steps:

a. Under **Host**, enter the IP address or FQDN of the vRLI host.
b. (Optional) Select the **Enable SSL?** checkbox to encrypt the logs being sent to vRLI using SSL.
c. Choose one of the following SSL certificate validation options:

- To skip certificate validation for the vRLI host, select the **Disable SSL certificate validation** checkbox. Select this option if you are using a self-signed certificate in order to simplify setup for a development or test environment.

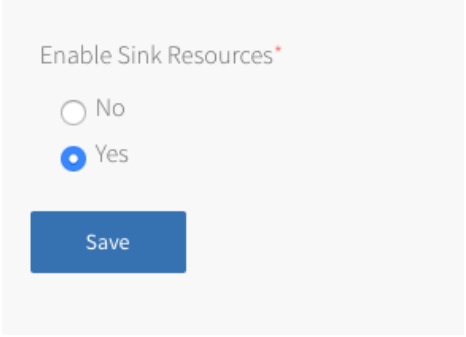
  Note: Disabling certificate validation is not recommended for production environments.

- To enable certificate validation for the vRLI host, clear the **Disable SSL certificate validation** checkbox.

d. (Optional) If your vRLI certificate is not signed by a trusted CA root or other well known certificate, enter the certificate in the **CA certificate** field. Locate the PEM of the CA used to sign the vRLI certificate, copy the contents of the certificate file, and paste them into the field. Certificates must be in PEM-encoded format.
e. Under **Rate limiting**, enter a time in milliseconds to change the rate at which logs are sent to the vRLI host. The rate limit specifies the minimum time between messages before the fluentd agent begins to drop messages. The default value (0) means the rate is not limited, which suffices for many deployments.

Note: If your deployment is generating a high volume of logs, you can increase this value to limit network traffic. Consider starting with a lower number, such as 10, and tuning to optimize for your deployment. A large number might result in dropping too many log entries.

8. To enable clusters to drain app logs to sinks using `syslog://`, select the **Enable Sink Resources** checkbox. For more information about using sink resources, see Creating Sink Resources.



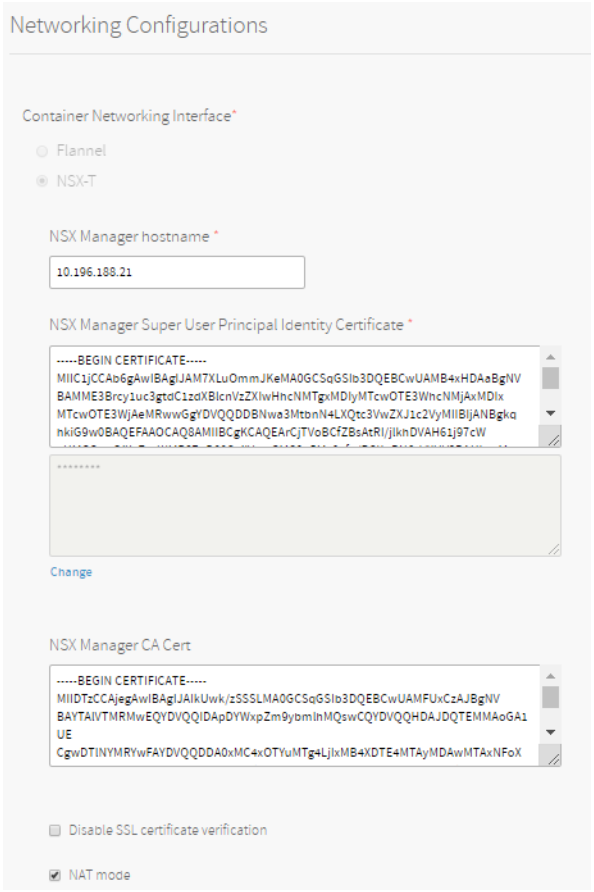
9. Click **Save**. These settings apply to any clusters created after you have saved these configuration settings and clicked **Apply Changes**. If the **Upgrade all clusters errand** has been enabled, these settings are also applied to existing clusters.

Note: The PKS tile does not validate your vRLI configuration settings. To verify your setup, look for log entries in vRLI.

Networking

To configure networking, do the following:

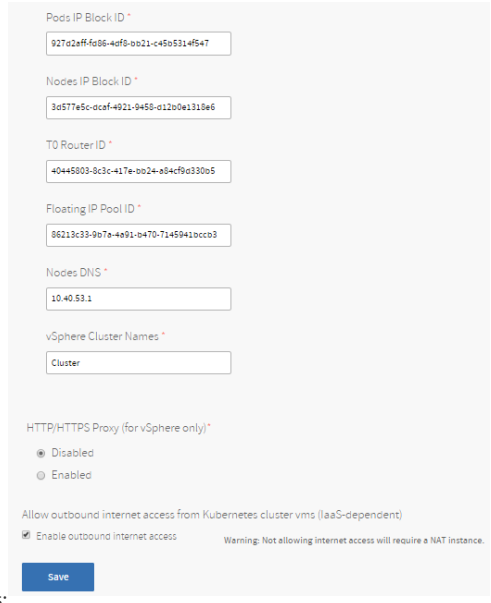
1. Click **Networking**.



2. Under **Container Networking Interface**, select **NSX-T**.
    a. For **NSX Manager hostname**, enter the hostname or IP address of your NSX Manager.
    b. For **NSX Manager Super User Principal Identify Certificate**, copy and paste the contents and private key of the Principal Identity certificate you created in Generating and Registering the NSX Manager Superuser Principal Identity Certificate and Key.
    c. For **NSX Manager CA Cert**, copy and paste the contents of the NSX Manager CA certificate you created in Generating and Registering the NSX Manager Certificate. Use this certificate and key to connect to the NSX Manager.
    d. The **Disable SSL certificate verification** checkbox is **not** selected by default. In order to disable TLS verification, select the checkbox. You may want to disable TLS verification if you did not enter a CA certificate, or if your CA certificate is self-signed.

> **Note**: The **NSX Manager CA Cert** field and the **Disable SSL certificate verification** option are intended to be mutually exclusive. If you disable SSL certificate verification, leave the CA certificate field blank. If you enter a certificate in the **NSX Manager CA Cert** field, do not disable SSL certificate verification. If you populate the certificate field and disable certificate validation, insecure mode takes precedence.

    e. If you are using a NAT deployment topology, leave the **NAT mode** checkbox selected. If you are using a No-NAT topology, clear this checkbox. For more information, see Deployment Topologies.



f. Enter the following IP Block settings:

- **Pods IP Block ID**: Enter the UUID of the IP block to be used for Kubernetes pods. PKS allocates IP addresses for the pods when they are created in Kubernetes. Each time a namespace is created in Kubernetes, a subnet from this IP block is allocated. The current subnet size that is created is /24, which means a maximum of 256 pods can be created per namespace.
- **Nodes IP Block ID**: Enter the UUID of the IP block to be used for Kubernetes nodes. PKS allocates IP addresses for the nodes when they are created in Kubernetes. The node networks are created on a separate IP address space from the pod networks. The current subnet size that is created is /24, which means a maximum of 256 nodes can be created per cluster. For more information, including sizes and the IP blocks to avoid using, see Plan IP Blocks in *Preparing NSX-T Before Deploying PKS*.

g. For **T0 Router ID**, enter the `t0-pks` T0 router UUID. Locate this value in the NSX-T UI router overview.
h. For **Floating IP Pool ID**, enter the `ip-pool-vips` ID that you created for load balancer VIPs. For more information, see Plan Network CIDRs. PKS uses the floating IP pool to allocate IP addresses to the load balancers created for each of the clusters. The load balancer routes the API requests to the master nodes and the data plane.
i. For **Nodes DNS**, enter one or more Domain Name Servers used by the Kubernetes nodes.
j. For **vSphere Cluster Names**, enter a comma-separated list of the vSphere clusters where you will deploy Kubernetes clusters. The NSX-T pre-check errand uses this field to verify that the hosts from the specified clusters are available in NSX-T. You can specify clusters in this format: `cluster1,cluster2,cluster3`.

3. (Optional) Configure a global proxy for all outgoing HTTP and HTTPS traffic from your Kubernetes clusters and the PKS API server. See Using Proxies with PKS on NSX-T for instructions on how to enable a proxy.

4. Under **Allow outbound internet access from Kubernetes cluster vms (IaaS-dependent)**, ignore the **Enable outbound internet access** checkbox.

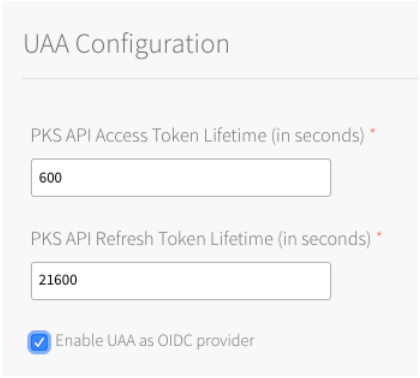
5. Click **Save**.

UAA

To configure the UAA server, do the following:

1. Click **UAA**.

2. Under **PKS API Access Token Lifetime**, enter a time in seconds for the PKS API access token lifetime.



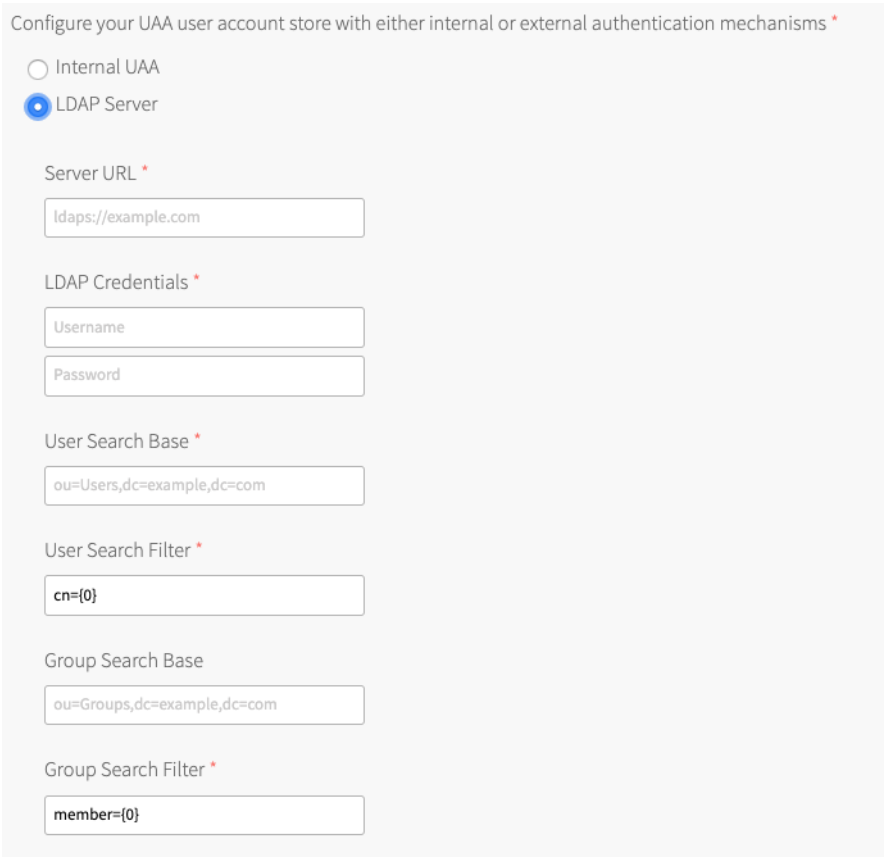
3. Under **PKS API Refresh Token Lifetime**, enter a time in seconds for the PKS API refresh token lifetime.
4. Select one of the following options:
    - To use an internal user account store for UAA, select **Internal UAA**. Click **Save** and continue to (Optional) Monitoring.
    - To use an external user account store for UAA, select **LDAP Server** and continue to Configure LDAP as an Identity Provider.

> **Note**: Selecting **LDAP Server** allows admin users to give cluster access to groups of users. For more information about performing this procedure, see Grant Cluster Access to a Group in *Managing Users in PKS with UAA*.

Configure LDAP as an Identity Provider

To integrate UAA with one or more LDAP servers, configure PKS with your LDAP endpoint information as follows:

1. Under **UAA**, select **LDAP Server**.



2. For **Server URL**, enter the URLs that point to your LDAP server. If you have multiple LDAP servers, separate their URLs with spaces. Each URL must include one of the following protocols:
    - `ldap://` : Use this protocol if your LDAP server uses an unencrypted connection.

- `ldaps://`: Use this protocol if your LDAP server uses SSL for an encrypted connection. To support an encrypted connection, the LDAP server must hold a trusted certificate or you must import a trusted certificate to the JVM truststore.

3. For **LDAP Credentials**, enter the LDAP Distinguished Name (DN) and password for binding to the LDAP server. For example, `cn=administrator,ou=Users,dc=example,dc=com`. If the bind user belongs to a different search base, you must use the full DN.

   Note: We recommend that you provide LDAP credentials that grant read-only permissions on the LDAP search base and the LDAP group search base.

4. For **User Search Base**, enter the location in the LDAP directory tree where LDAP user search begins. The LDAP search base typically matches your domain name.

   For example, a domain named `cloud.example.com` may use `ou=Users,dc=example,dc=com` as its LDAP user search base.

5. For **User Search Filter**, enter a string to use for LDAP user search criteria. The search criteria allows LDAP to perform more effective and efficient searches. For example, the standard LDAP search filter `cn=Smith` returns all objects with a common name equal to `Smith`.

   In the LDAP search filter string that you use to configure PKS, use `{0}` instead of the username. For example, use `cn={0}` to return all LDAP objects with the same common name as the username.

   In addition to `cn`, other common attributes are `mail`, `uid` and, in the case of Active Directory, `sAMAccountName`.

   Note: For information about testing and troubleshooting your LDAP search filters, see Configuring LDAP Integration with Pivotal Cloud Foundry.

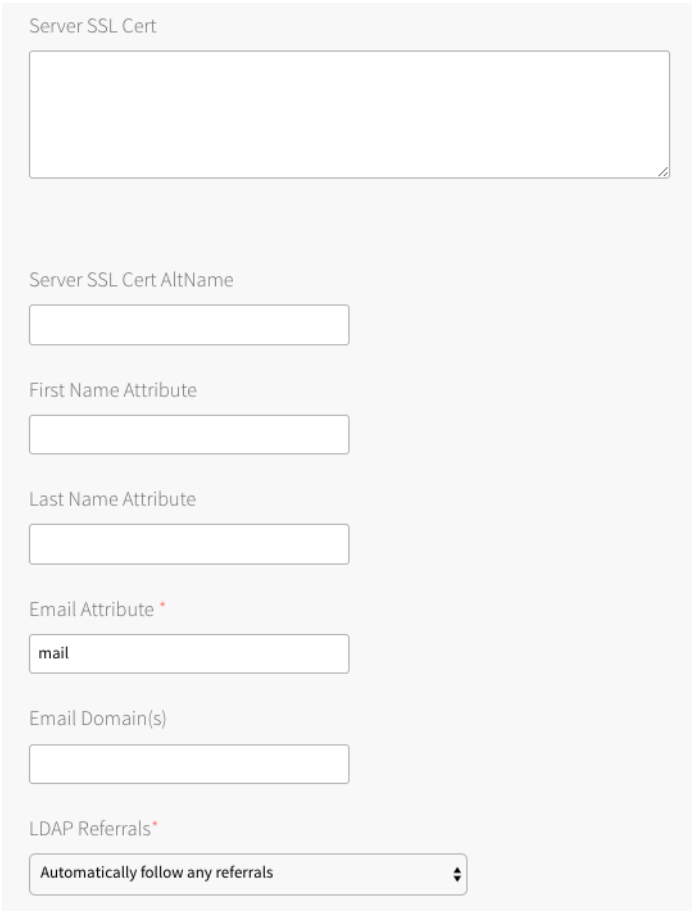
6. For **Group Search Base**, enter the location in the LDAP directory tree where the LDAP group search begins.

   For example, a domain named `cloud.example.com` may use `ou=Groups,dc=example,dc=com` as its LDAP group search base.

   Follow the instructions in the Grant PKS Access to an External LDAP Group section of *Managing Users in PKS with UAA* to map the groups under this search base to roles in PKS.

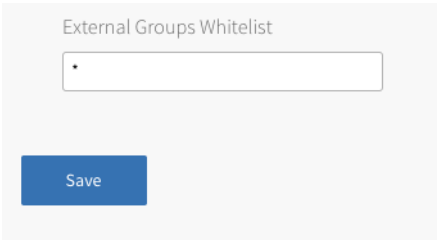
7. For **Group Search Filter**, enter a string that defines LDAP group search criteria. The standard value is `member={0}`.

8. For **Server SSL Cert**, paste in the root certificate from your CA certificate or your self-signed certificate.



9. For **Server SSL Cert AltName**, do one of the following:
    - If you are using `ldaps://` with a self-signed certificate, enter a Subject Alternative Name (SAN) for your certificate.
    - If you are not using `ldaps://` with a self-signed certificate, leave this field blank.
10. For **First Name Attribute**, enter the attribute name in your LDAP directory that contains user first names. For example, `cn`.
11. For **Last Name Attribute**, enter the attribute name in your LDAP directory that contains user last names. For example, `sn`.
12. For **Email Attribute**, enter the attribute name in your LDAP directory that contains user email addresses. For example, `mail`.
13. For **Email Domain(s)**, enter a comma-separated list of the email domains for external users who can receive invitations to Apps Manager.
14. For **LDAP Referrals**, choose how UAA handles LDAP server referrals to other user stores. UAA can follow the external referrals, ignore them without returning errors, or generate an error for each external referral and abort the authentication.
15. For **External Groups Whitelist**, enter a comma-separated list of group patterns which need to be populated in the user's `id_token`. For further information on accepted patterns see the description of the `config.externalGroupsWhitelist` in the OAuth/OIDC Identity Provider Documentation.

    Note: When sent as a Bearer token in the Authentication header, wide pattern queries for users who are members of multiple groups, can cause the size of the `id_token` to extend beyond what is supported by web servers.

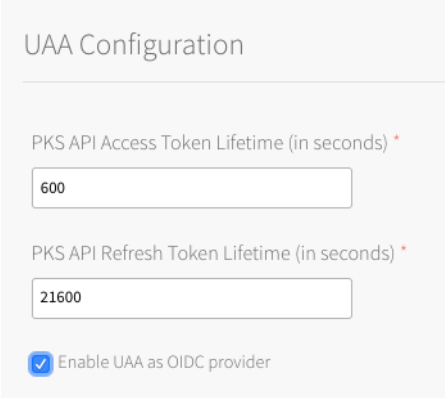


16. Click **Save**.

(Optional) Configure OpenID Connect

You can use OpenID Connect (OIDC) to instruct Kubernetes to verify end-user identities based on authentication performed by an authorization server, such as UAA.

To configure PKS to use OIDC, select **Enable UAA as OIDC provider**. With OIDC enabled, Admin Users can grant cluster-wide access to Kubernetes end users.



For more information about configuring OIDC, see the table below:

Option	Description
OIDC disabled	If you do not enable OIDC, Kubernetes authenticates users against its internal user management system.
OIDC enabled	If you enable OIDC, Kubernetes uses the authentication mechanism that you selected in UAA as follows: • If you selected **Internal UAA**, Kubernetes authenticates users against the internal UAA authentication mechanism. • If you selected **LDAP Server**, Kubernetes authenticates users against the LDAP server.

For additional information about getting credentials with OIDC configured, see Retrieve Cluster Credentials in *Retrieving Cluster Credentials and Configuration*.

Note: When you enable OIDC, existing PKS-provisioned Kubernetes clusters are upgraded to use OIDC. This invalidates your kubeconfig files. You must regenerate the files for all clusters.

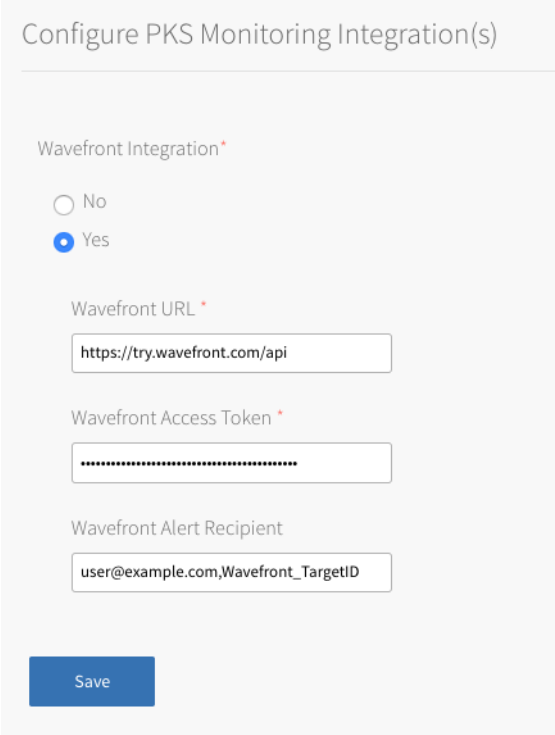
(Optional) Monitoring

You can monitor Kubernetes clusters and pods metrics externally using the integration with Wavefront by VMware.

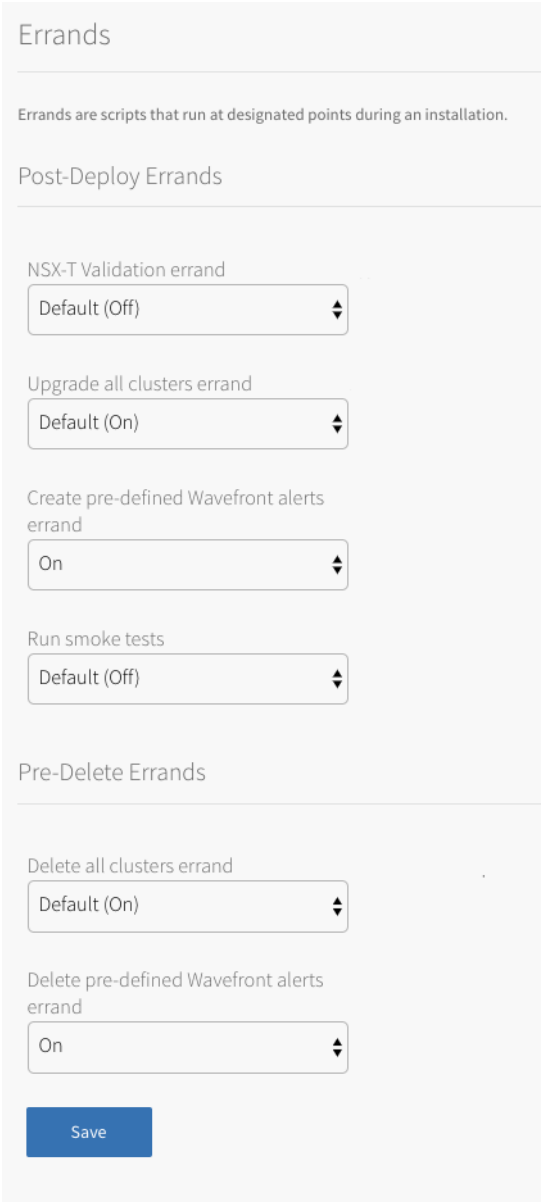
Note: Before you configure Wavefront integration, you must have an active Wavefront account and access to a Wavefront instance. You provide your Wavefront access token during configuration and enabling errands. For additional information, see Pivotal Container Service Integration Details in the Wavefront documentation.

By default, monitoring is disabled. To enable and configure Wavefront monitoring, do the following:

1. Select **Monitoring**.



2. On the **Monitoring** pane, under **Wavefront Integration**, select **Yes**.
3. Under **Wavefront URL**, enter the URL of your Wavefront subscription. For example, `https://try.wavefront.com/api`.
4. Under **Wavefront Access Token**, enter the API token for your Wavefront subscription.
5. To configure Wavefront to send alerts by email, enter email addresses or Wavefront Target IDs separated by commas under **Wavefront Alert Recipient**. For example, `user@example.com,Wavefront_TargetID`. To create alerts, you must enable errands.
6. Select **Errands**.
7. On the **Errands** pane, enable **Create pre-defined Wavefront alerts errand**.



8. Enable **Delete pre-defined Wavefront alerts errand**.
9. Click **Save**. Your settings apply to any clusters created after you have saved these configuration settings and clicked **Apply Changes**.

> **Note**: The PKS tile does not validate your Wavefront configuration settings. To verify your setup, look for cluster and pod metrics in Wavefront.

Usage Data

VMware's Customer Experience Improvement Program (CEIP) and the Pivotal Telemetry Program (Telemetry) provides VMware and Pivotal with information that enables the companies to improve their products and services, fix problems, and advise you on how best to deploy and use our products. As part of the CEIP and Telemetry, VMware and Pivotal collect technical information about your organization's use of the Pivotal Container Service (PKS) on a regular basis. Since PKS is jointly developed and sold by VMware and Pivotal, we will share this information with one another. Information collected under CEIP or Telemetry does not personally identify any individual.

Regardless of your selection in the **Usage Data** pane, a small amount of data is sent from Cloud Foundry Container Runtime (CFCR) to the PKS tile. However, that data is not shared externally.

To configure the **Usage Data** pane, perform the following steps:

1. Select the **Usage Data** side-tab.
2. Read the Usage Data description.

3. Make your selection.

    a. To join the program, select **Yes, I want to join the CEIP and Telemetry Program for PKS**.
    b. To decline joining the program, select **No, I do not want to join the CEIP and Telemetry Program for PKS**.

4. Click **Save**.

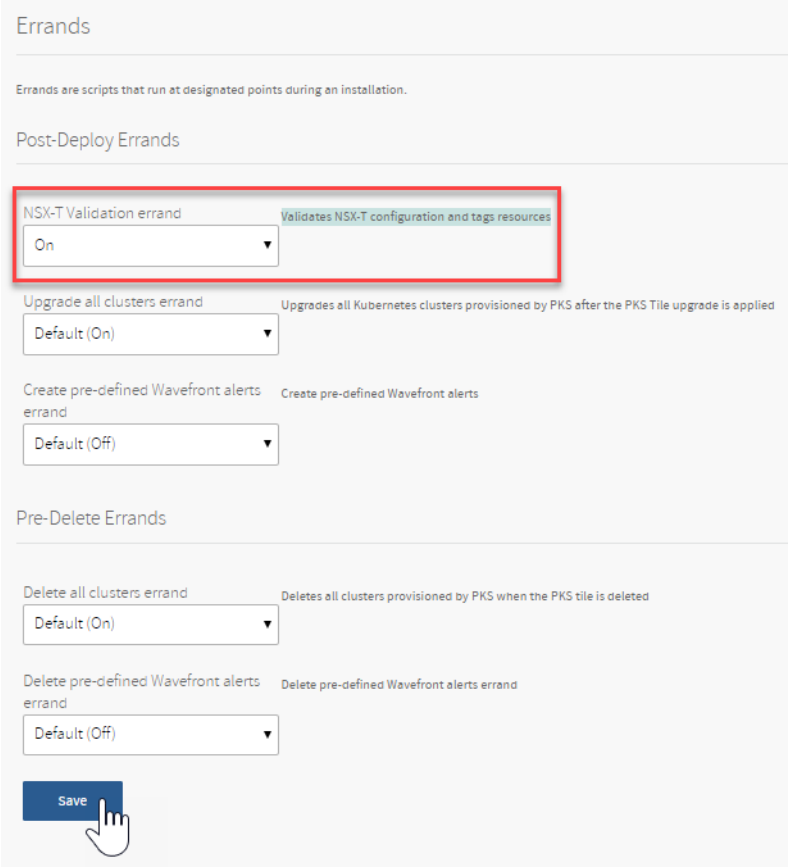
Note: If you join the CEIP and Telemetry Program for PKS, open your firewall to allow outgoing access to `https://vcsa.vmware.com/ph-prd` on port `443`.

Errands

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

To configure when post-deploy and pre-delete errands for PKS are run, make a selection in the dropdown next to the errand.

WARNING: You must enable the NSX-T Validation errand to verify and tag required NSX-T objects.



For more information about errands and their configuration state, see Managing Errands in Ops Manager.

WARNING: Because PKS uses floating stemcells, updating the PKS tile with a new stemcell triggers the rolling of every VM in each cluster. Also, updating other product tiles in your deployment with a new stemcell causes the PKS tile to roll VMs. This rolling is enabled by the **Upgrade all clusters errand**. We recommend that you keep this errand turned on because automatic rolling of VMs ensures that all deployed cluster VMs are patched. However, automatic rolling can cause downtime in your deployment.

Resource Config

VMs used by **Pivotal Container Service** jobs must meet the following minimum requirements:

CPU	Memory	Disk

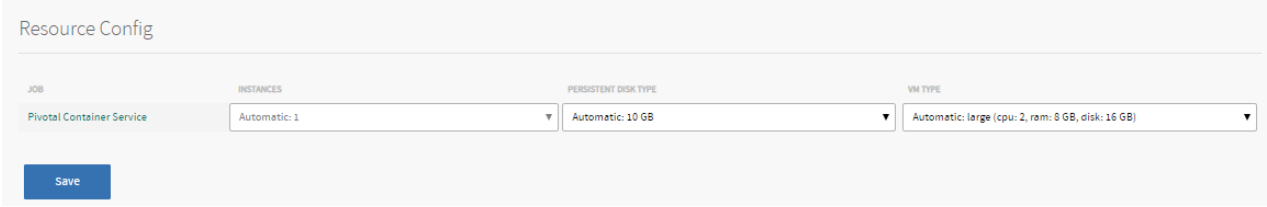
2	8 GB	29 GB

Note: If you experience timeouts or slowness when interacting with the PKS API, select a **VM Type** with greater CPU and memory resources.

To deploy **Pivotal Container Service** job VMs meeting the minimum requirements, perform the following steps:

1. Click **Resource Config**.

   The default "Automatic" **VM Type** does not meet the **Pivotal Container Service** job VM minimum requirements:

2. Select a **VM Type** with CPU, memory and disk resources either matching or exceeding the minimum **Pivotal Container Service** job VM requirements.

3. Select **Save**.

Step 3: Apply Changes

After configuring the PKS tile, follow the steps below to deploy the tile:

1. Return to the Ops Manager Installation Dashboard.

2. Click **Review Pending Changes**. Select the product that you intend to deploy and review the changes. For more information, see Reviewing Pending Product Changes.

3. Click **Apply Changes**.

Step 4: Install the PKS and Kubernetes CLIs

The PKS and Kubernetes CLIs help you interact with your PKS-provisioned Kubernetes clusters and Kubernetes workloads. To install the CLIs, follow the instructions below:

- Installing the PKS CLI
- Installing the Kubernetes CLI

Step 5: Verify NAT Rules

If you are using NAT mode, verify that you have created the required NAT rules for the PKS Management Plane. See Creating the PKS Management Plane for details.

In addition, for NAT and no-NAT modes, verify that you created the required NAT rule for Kubernetes master nodes to access NSX Manager. See Prepare Compute Plane for details.

Lastly, if you want your developers to be able to access the PKS CLI from their external workstations, create a DNAT rule that maps a routable IP address to the PKS API VM. This must be done after PKS is successfully deployed and it has an IP address. See Create DNAT Rule on T0 Router for External Access to the PKS CLI for details.

Step 6: Configure PKS API Access

Follow the procedures in Configuring PKS API Access.

Step 7: Configure Authentication for PKS

Configure authentication for PKS using User Account and Authentication (UAA). For information, see Managing Users in PKS with UAA.

Next Steps

After installing PKS on vSphere with NSX-T integration, you may want to do one or more of the following:

- Integrate VMware Harbor with PKS to store and manage container images. For more information, see Integrating VMware Harbor Registry with PKS.
- Create your first PKS cluster. For more information, see Creating Clusters.

Please send any feedback you have to pks-feedback@pivotal.io.

Implementing a Multi-Foundation PKS Deployment

Page last updated:

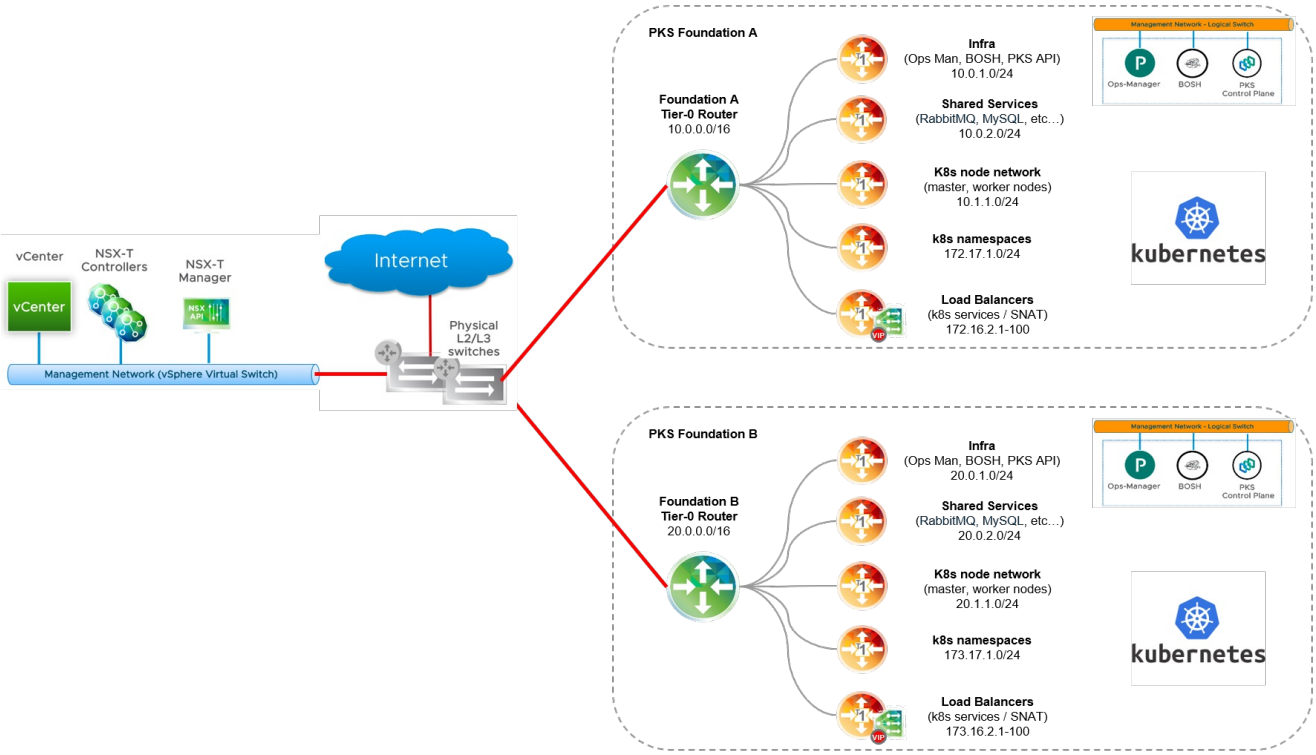
This topic describes how to deploy multiple instances of PKS on vSphere with NSX-T infrastructure.

About Multi-Foundation PKS

A multi-foundation deployment of PKS lets you install and run multiple instances of PKS. The purpose of a multi-foundation deployment of PKS is to share a common vSphere and NSX-T infrastructure across multiple foundations, while providing complete networking isolation across foundations.

As shown in the diagram, with a multi-foundation PKS topology, each PKS instance is deployed to a dedicated NSX-T Tier-0 router. Foundation A T0 router with Management CIDR 10.0.0.0/16 connects to the vSphere and NSX-T infrastructure. Similarly, Foundation B T0 router with Management CIDR 20.0.0.0/16 connects to the same vSphere and NSX-T components.

As with a single instance deployment, PKS management components are deployed to a dedicated network, for example, 10.0.0.0/24 for PKS Foundation A; 20.0.0.0/24 for PKS Foundation B. When PKS is deployed, networks are defined for nodes, pods, and load balancers. Because of the dedicated Tier-0 router, there is complete networking isolation between each PKS instance.



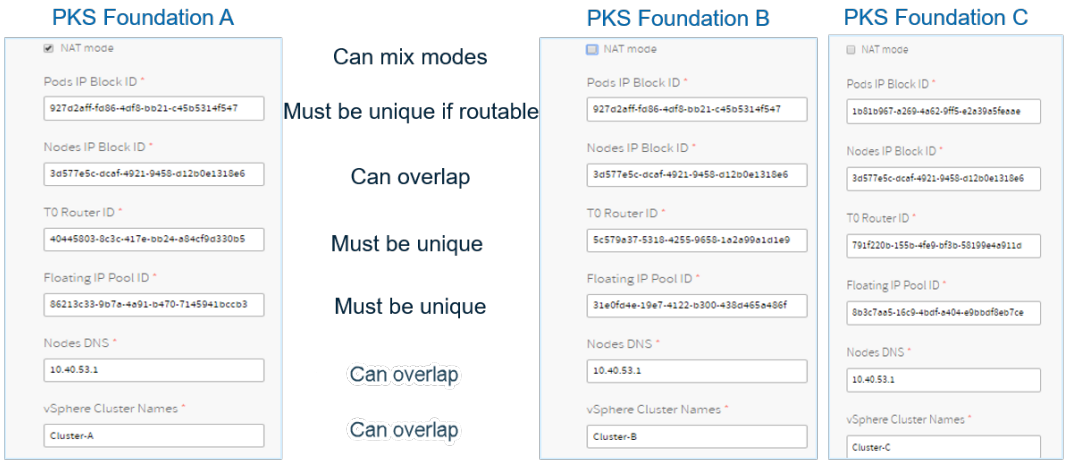
Requirements

To implement a multi-foundation PKS topology, adhere to the following requirements:

- One Tier-0 router for each PKS instance. For more information, see Configuring Multiple Tier-0 Routers for Tenant Isolation.
- The Floating IP pool must not overlap. The CIDR range for each Floating IP Pool must be unique and not overlapping across foundations. For more information, see Create Floating IP Pool.
- PKS instances can be deployed in NAT and no-NAT mode. If more than one PKS instance is deployed in no-NAT mode, the Nodes IP Block networks cannot overlap.
- For any Pods IP Block used to deploy Kubernetes clusters in no-NAT (routable) mode, the Pods IP Block cannot overlap across foundations.
- The NSX-T Super User Principal Identity Certificate should be unique per PKS instance.

The image below shows three PKS installations across three Tier-0 foundations. Key considerations to keep in mind with a multi-foundation PKS topology include the following:

- Each foundation must rely on a dedicated Tier-0 router
- You can mix-and-match NAT and no-NAT mode across foundations for Node and Pod networks
- If you are using non-routable Pods IP Block networks, the Pods IP Block addresses can overlap across foundations
- Because Kubernetes nodes are behind a dedicated Tier-0 router, if clusters are deployed in NAT mode the Nodes IP Block addresses can also overlap across foundations
- For each foundation you must define a unique Floating ID Pool with non-overlapping IPs



Please send any feedback you have to pks-feedback@pivotal.io.

Using Proxies with PKS on NSX-T

This topic describes how to use proxies with Pivotal Container Service (PKS) with NSX-T.

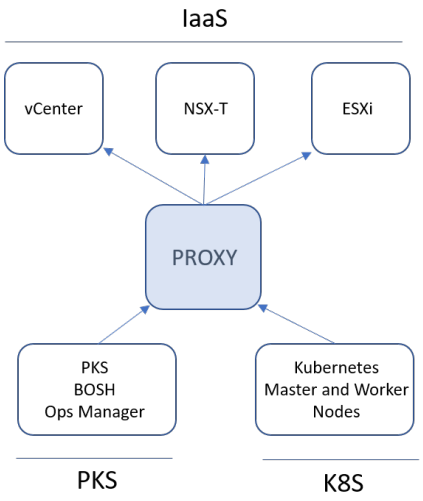
Overview

If your environment includes HTTP proxies, you can configure PKS with NSX-T to use these proxies so that PKS-deployed Kubernetes master and worker nodes access public Internet services and other internal services through a proxy.

In addition, PKS proxy settings apply to the PKS API instance. When a PKS operator creates a Kubernetes cluster, the PKS API instance VM behind a proxy is able to manage NSX-T objects on the standard network.

You can also proxy outgoing HTTP/HTTPS traffic from Ops Manager and the BOSH Director so that all PKS components use the same proxy service.

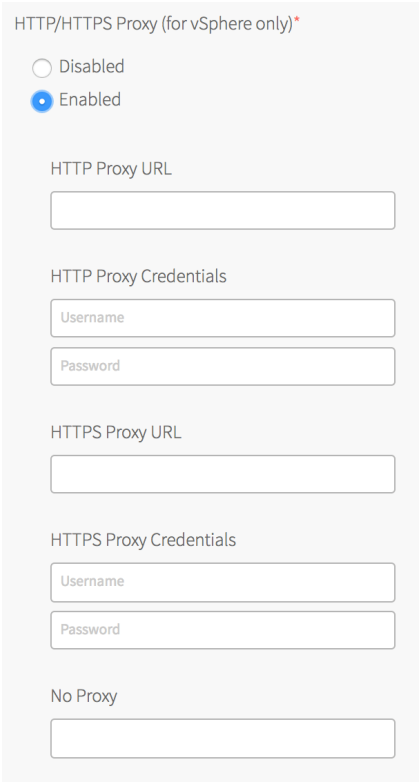
The following diagram illustrates the network architecture:



Enable PKS API and Kubernetes Proxy

To configure a global HTTP proxy for all outgoing HTTP/HTTPS traffic from the Kubernetes cluster nodes and the PKS API server, perform the following steps:

1. Navigate to Ops Manager and log in.
2. Click the PKS tile.
3. Click **Networking**.



4. Under **HTTP/HTTPS proxy**, select **Enabled**. When this option is enabled, you can proxy HTTP traffic, HTTPS traffic, or both.
5. To proxy outgoing HTTP traffic, under **HTTP Proxy URL**, enter the HTTP URL of your proxy endpoint. For example, `http://myproxy.com:80`.
6. If the proxy for outgoing HTTP traffic uses basic authentication, enter the user name and password in the **HTTP Proxy Credentials** fields.
7. To proxy outgoing HTTPS traffic, under **HTTPS Proxy URL**, enter the HTTP URL of your proxy endpoint. For example, `http://myproxy.com:80`.

   Note: Using an HTTPS connection to the proxy server is not supported. HTTP and HTTPS proxy options can only be configured with an HTTP connection to the proxy server. You cannot populate either of the proxy URL fields with an HTTPS URL. The proxy host and port can be different for HTTP and HTTPS traffic, but the proxy protocol must be HTTP.

8. If the proxy for outgoing HTTPS traffic uses basic authentication, enter the user name and password in the **HTTPS Proxy Credentials** fields.
9. Under **No Proxy**, enter the comma-separated list of IP addresses that must bypass the proxy to allow for internal PKS communication.

   In addition to `127.0.0.1` and `localhost`, you must include your deployment network CIDR, your node network IP block, and your pod network IP block CIDR:

   ```
   127.0.0.1,localhost,
   DEPLOYMENT-NETWORK-CIDR,
   NODE-NETWORK-IP-BLOCK-CIDR,
   POD-NETWORK-IP-BLOCK-CIDR
   ```

   You can enter FQDNs in the **No Proxy** field. For example:

   - `registry-1.docker.io`
   - `auth.docker.io`
   - `production.cloudflare.docker.com`
   - `gcr.io`
   - `storage.googleapis.com`

   If you are upgrading and have an existing proxy configuration for reaching a Docker registry or other external services, add the following IP addresses to the **No Proxy** field to prevent the PKS to IaaS traffic from going through the proxy: NSX Manager, vCenter Server, and all ESXi hosts.

   If a component is communicating with PKS or Harbor using a hostname instead of an IP address, you will need to add the corresponding FQDN to the **No Proxy** list. For example:

```
127.0.0.1,localhost,
DEPLOYMENT-NETWORK-CIDR,
NODE-NETWORK-IP-BLOCK-CIDR,
POD-NETWORK-IP-BLOCK-CIDR,
PKS-API-FQDN,HARBOR-API-FQDN
```

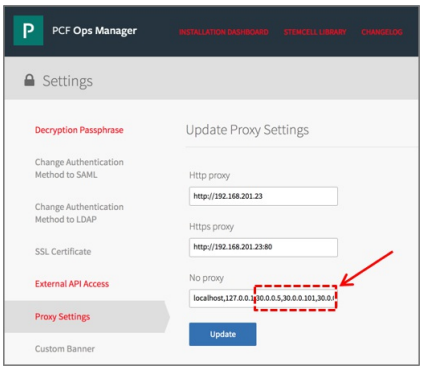
Note: By default, the `.internal`, `10.100.0.0/8`, and `10.200.0.0/8` IP address ranges are not proxied. This allows internal PKS communication.

10. Save the changes to the PKS tile.
11. Proceed with any remaining PKS tile configurations and deploy PKS. See Installing PKS on vSphere with NSX-T.

Enable Ops Manager and BOSH Proxy

To enable an HTTP proxy for outgoing HTTP/HTTPS traffic from Ops Manager and the BOSH Director, perform the following steps:

1. Navigate to Ops Manager and log in.
2. Select **User Name > Settings** in the upper right.
3. Click **Proxy Settings**.



4. Under **HTTP Proxy**, enter the FQDN or IP address of the HTTP proxy endpoint. For example, `http://myproxy.com:80`.
5. Under **HTTPS Proxy**, enter the FQDN or IP address of the HTTPS proxy endpoint. For example, `http://myproxy.com:80`.

Note: Using an HTTPS connection to the proxy server is not supported. Ops Manager and BOSH HTTP and HTTPS proxy options can be only configured with an HTTP connection to the proxy.

6. Under **No Proxy**, include the hosts that must bypass the proxy. This is required.

   In addition to `127.0.0.1` and `localhost`, include the BOSH Director IP and the PKS VM IP. The BOSH Director IP is typically the first IP address in the deployment network CIDR, and the PKS VM IP is the second IP address in the deployment network CIDR. In addition, be sure to include the Ops Manager IP address in the **No Proxy** field as well.

```
127.0.0.1,localhost,BOSH-DIRECTOR-IP,PKS-VM-IP,OPS-MANAGER-IP
```

Note: Ops Manager does not allow the use of a CIDR range in the **No Proxy** field. You must specify each individual IP address to bypass the proxy.

The **No Proxy** field does not accept wildcard domain notation, such as `*.docker.io` and `*.docker.com`. You must specify the exact IP or FQDN to bypass the proxy, such as `registry-1.docker.io`.

7. Click **Save**.
8. Return to the Ops Manager Installation Dashboard and click **Review Pending Changes**.
9. Click **Apply Changes** to deploy Ops Manager and the BOSH Director with the updated proxy settings.

Please send any feedback you have to pks-feedback@pivotal.io.

Defining Network Profiles

Page last updated:

This topic describes how to define network profiles for Kubernetes clusters provisioned with Pivotal Container Service (PKS) on vSphere with NSX-T.

About Network Profiles

Network profiles let you customize NSX-T configuration parameters at the time of cluster creation. Use cases for network profiles include the following:

Profile Type	Description
Load Balancer Sizing	Customize the size of the NSX-T load balancer provisioned when a Kubernetes cluster is created using PKS.
Custom Pod Networks	Assign IP addresses from a dedicated IP block to pods in your Kubernetes cluster.
Routable Pod Networks	Assign routable IP addresses from a dedicated IP block to pods in your Kubernetes cluster.
Bootstrap Security Group for Kubernetes Master Nodes	Specify an NSX-T Namespace Group where Kubernetes master nodes will be added to during cluster creation.
Pod Subnet Prefix	Specify the size of the pod subnet.
Custom Floating IP	Specify a custom floating IP pool.
Edge Router Selection	Specify the NSX-T Tier-0 router where Kubernetes node and Pod networks will be connected to.
DNS Configuration for Kubernetes Clusters	Specify one or more DNS servers for Kubernetes clusters.

Network Profile Format

Network profiles are defined using JSON. Here are example network profiles for two different customers:

```
np_customer_A.json
{
   "name": "np-cust-a",
   "description": "Network Profile for Customer A",
   "parameters": {
      "lb_size": "small",
      "t0_router_id": "5a7a82b2-37e2-4d73-9cb1-97a8329e1a90",
      "fip_pool_ids": [
         "e50e8f6e-1a7a-45dc-ad49-3a607baa7fa0"
      ],
      "pod_ip_block_ids": [
         "7056d707-acec-470e-88cf-66bb86fbf439"
      ],
      "master_vms_nsgroup_id": "9b8d535a-d3b6-4735-9fd0-56305c4a5293",
      "pod_subnet_prefix" : 27
   }
}
```

```
np_customer_B.json
{
   "name": "np-cust-b",
   "description": "Network Profile for Customer B",
   "parameters": {
      "lb_size": "medium",
      "t0_router_id": "5a7a82b2-37e2-4d73-9cb1-97a8329e1a92",
      "fip_pool_ids": [
         "e50e8f6e-1a7a-45dc-ad49-3a607baa7fa2"
      ],
      "pod_routable": true,
      "pod_ip_block_ids": [
         "ebe78a74-a5d5-4dde-ba76-9cf4067eee55",
         "ebe78a74-a5d5-4dde-ba76-9cf4067eee56"
      ]
      "master_vms_nsgroup_id": "9b8d535a-d3b6-4735-9fd0-56305c4a5292",
      "pod_subnet_prefix" : 26
   }
}
```

Network Profile Parameters

Define a network profile configuration in a JSON file using the following parameters:

Parameter	Description
`name`	String that is the user-defined name of the network profile.
`description`	String that is the user-defined description for the network profile.
`parameters`	One or more name-value pairs.
`lb_size`	Size of the NSX-T load balancer deployed with the Kubernetes cluster: `small`, `medium`, or `large`.
`pod_ip_block_ids`	Array of Pod IP Block UUIDs as defined in NSX-T; comma-separated.
`pod_routable`	Boolean `true` or `false`. Set the parameter to `true` to assign routable IP addresses to pods.
`master_vms_nsgroup_id`	UUID of NSGroup as defined in NSX-T.
`fip_pool_ids`	Array of Floating IP Pool UUIDs as defined in NSX-T; comma separated.
`pod_subnet_prefix`	Integer that is the prefix size of the custom Pods IP Block subnet.
`t0_router_id`	UUID of tenant Tier-0 router as defined in NSX-T.
`nodes_dns`	Array of IP addresses (up to 3) for DNS server lookup by Kubernetes nodes and pods.

Network Profile Creation

After the network profile is defined in a JSON file, a PKS administrator can create the network profile using the PKS CLI. The Kubernetes administrator can use the network profile when creating a cluster.

For more information, see the Create and Use Network Profiles section of *Using Network Profiles (NSX-T Only)*.

Load Balancer Sizing

When you deploy a Kubernetes cluster using PKS on NSX-T, an NSX-T load balancer is automatically provisioned. By default the size of this load balancer is small. Using a network profile, you can customize the size of the load balancer. For more information, see Load Balancers in PKS Deployments on vSphere with NSX–T.

NSX-T load balancers run on edge nodes. There are various form factors for edge nodes. PKS supports the large edge VM and the bare metal edge. The large VM edge node must run on Intel processors. The large load balancer requires a bare metal edge node. For more information about edge nodes, see Scaling Load Balancer Resources in the NSX-T documentation.

The NSX-T load balancer is a logical load balancer that handles a number of functions using virtual servers and pools. For more information, see Supported Load Balancer Features in the NSX-T documentation.

The following virtual servers are required for PKS:

- 1 TCP layer 4 virtual server for each Kubernetes service of `type:LoadBalancer`
- 2 HTTP and HTTPS layer 7 global virtual servers for Kubernetes ingress resources
- 1 global virtual server for the PKS API

The following network profile, `np-lb-med`, defines a medium load balancer:

```
{
  "name": "np-lb-med",
  "description": "Network profile for medium NSX-T load balancer",
  "parameters": {
    "lb_size": "medium"
  }
}
```

The following network profile, `np-lb-large`, defines a large load balancer:

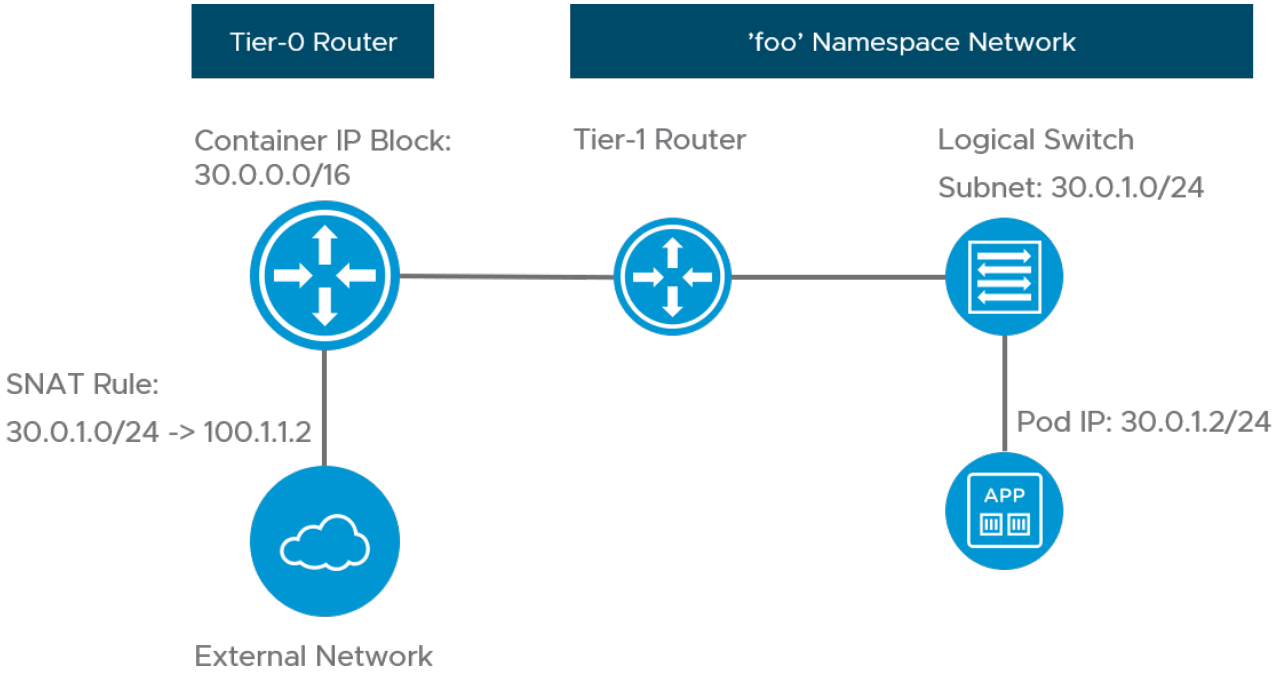
```
{
  "name": "np-lb-large",
  "description": "Network profile for large NSX-T load balancer",
  "parameters": {
    "lb_size": "large"
  }
}
```

Note: The large load balancer requires a bare metal NSX Edge Node.

Custom Pod Networks

When you configure your NSX-T infrastructure for PKS, you must create a **Pods IP Block**. For more information, see the Plan IP Blocks section of *Planning, Preparing, and Configuring NSX-T for PKS*.

By default, this subnet is non-routable. When a Kubernetes cluster is deployed, each pod receives an IP address from the **Pods IP Block** you created. Because the pod IP addresses are non-routable, NSX-T creates a SNAT rule on the Tier-0 router to allow network egress from the pods. This configuration is shown in the diagram below:



You can use a network profile to override the global **Pods IP Block** that you specify in the PKS tile with a custom IP block. To use a custom pods network, do the following after you deploy PKS:

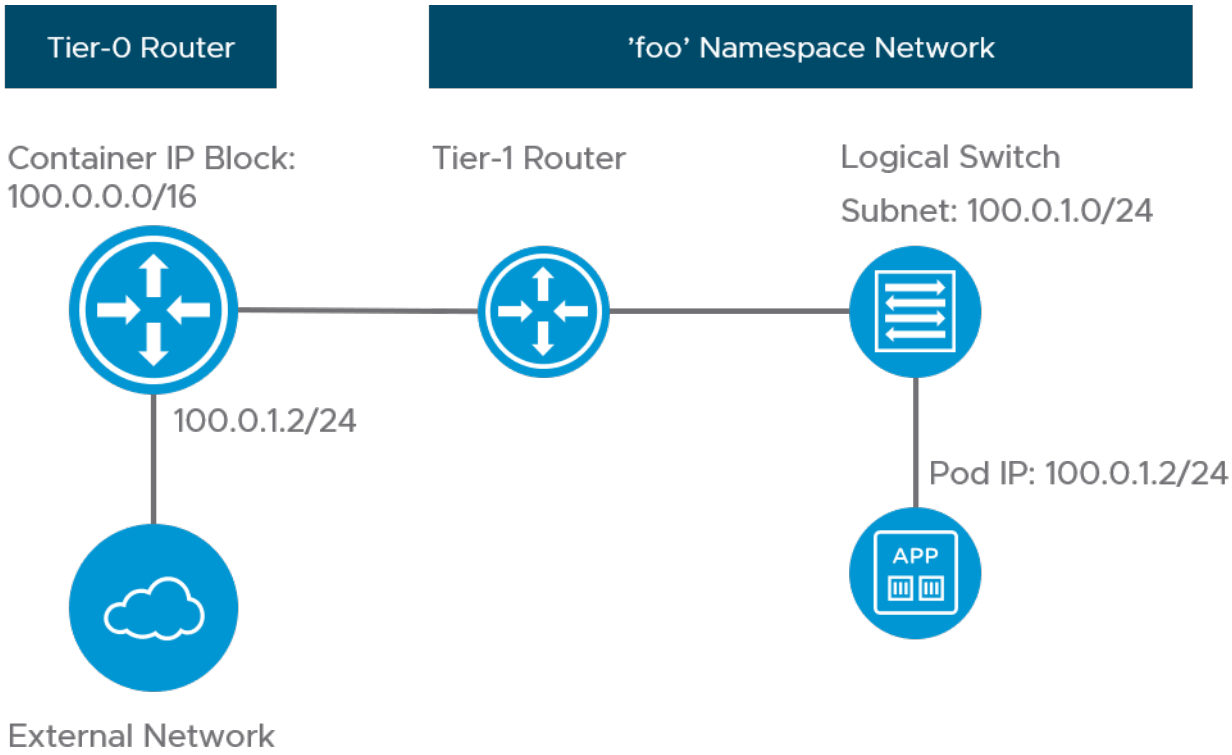
1. Define a custom IP block in NSX-T. For more information, see Creating NSX-T Objects for PKS.

2. Define a network profile that references the custom pods IP block. For example, the following network profile defines non-routable pod addresses from two IP blocks:

```
{
  "description": "Network profile with two non-routable pod networks",
  "name": "non-routable-pod",
  "parameters": {
    "pod_ip_block_ids": [
      "ebe78a74-a5d5-4dde-ba76-9cf4067eee55",
      "ebe78a74-a5d5-4dde-ba76-9cf4067eee56"
    ]
  }
}
```

Routable Pod Networks

Using a network profile, you can assign routable IP addresses from a dedicated routable IP block to pods in your Kubernetes cluster. When a cluster is deployed using that network profile, the routable IP block overrides the default non-routable IP block described created for deploying PKS. When you deploy a Kubernetes cluster using that network profile, each pod receives a routable IP address. This configuration is shown in the diagram below. If you use routable pods, the SNAT rule is not created.



To use routable pods, do the following after you deploy PKS:

1. Define a routable IP block in NSX-T. For more information, see Creating NSX-T Objects for PKS.

2. Define a network profile that references the routable IP block. For example, the following network profile defines routable pod addresses from two IP blocks:

```
{
  "description": "Network profile with small load balancer and two routable pod networks",
  "name": "small-routable-pod",
  "parameters": {
    "pod_routable": true,
    "pod_ip_block_ids": [
      "ebe78a74-a5d5-4dde-ba76-9cf4067eee55",
      "ebe78a74-a5d5-4dde-ba76-9cf4067eee56"
    ]
  }
}
```

Bootstrap Security Group

Most of the NSX-T virtual interface tags used by PKS are added to the Kubernetes master node or nodes during the node initialization phase of cluster provisioning. To add tags to virtual interfaces, the Kubernetes master node needs to connect to the NSX-T Manager API. Network security rules provisioned prior to cluster creation time do not allow nodes to connect to NSX-T if the rules are based on a Namespace Group (NSGroup) managed by PKS.

To address this bootstrap issue, PKS exposes an optional configuration parameter in Network Profiles to systematically add Kubernetes master nodes to a pre-provisioned NSGroup. The BOSH vSphere cloud provider interface (CPI) has the ability to use the NSGroup to automatically manage members following the BOSH VM lifecycle for Kubernetes master nodes.

To configure a Bootstrap Security Group, complete the following steps:

1. Create the NSGroup in NSX Manager prior to provisioning a Kubernetes cluster using PKS. For more information, see Create an NSGroup in the NSX-T documentation.

2. Define a network profile that references the NSGroup UUID that the BOSH CPI can use to bootstrap the master node or nodes. For example, the following network profile specifies an NSGroup for the BOSH CPI to use to dynamically update Kubernetes master node memberships:

```
{
  "name": "np-boot-nsgroups",
  "description": "Network Profile for Customer B",
  "parameters": {
    "master_vms_nsgroup_id": "9b8d535a-d3b6-4735-9fd0-56305c4a5293"
  }
}
```

Pod Subnet Prefix

Each time a Kubernetes namespace is created, a subnet from the pods IP block is allocated. The size of the subnet carved from this block for such purposes is /24. For more information, see the Pods IP Block section of *Planning, Preparing, and Configuring NSX-T for PKS*.

You can define a Network Profile using the `pod_subnet_prefix` parameter to customize the size of the pod subnet reserved for namespaces. For example, the following network profile specifies /27 for the size of the pods IP block subnet:

```
{
  "name": "np-pod-prefix",
  "description": "Network Profile for Customizing Pod Subnet Size",
  "parameters": {
    "pod_subnet_prefix" : 27
  }
}
```

Custom Floating IP Pool

To deploy PKS to vSphere with NSX-T, you must define a floating IP pool in NSX Manager. The IP addresses in this floating IP pool are assigned to load balancers automatically provisioned by NSX-T when you deploy a Kubernetes cluster using PKS. For more information, see the Plan Network CIDRs section of *Planning, Preparing, and Configuring NSX-T for PKS*.

You can define a network profile that specifies a custom floating IP pool to use instead of the default pool specified in the PKS tile.

To define a custom floating IP pool, follow the steps below:

1. Create a floating IP pool using NSX Manager prior to provisioning a Kubernetes cluster using PKS. For more information, see Create IP Pool in the NSX-T documentation.

2. Define a network profile that references the floating IP pool UUID that you defined. The following example defines a custom floating IP pool:

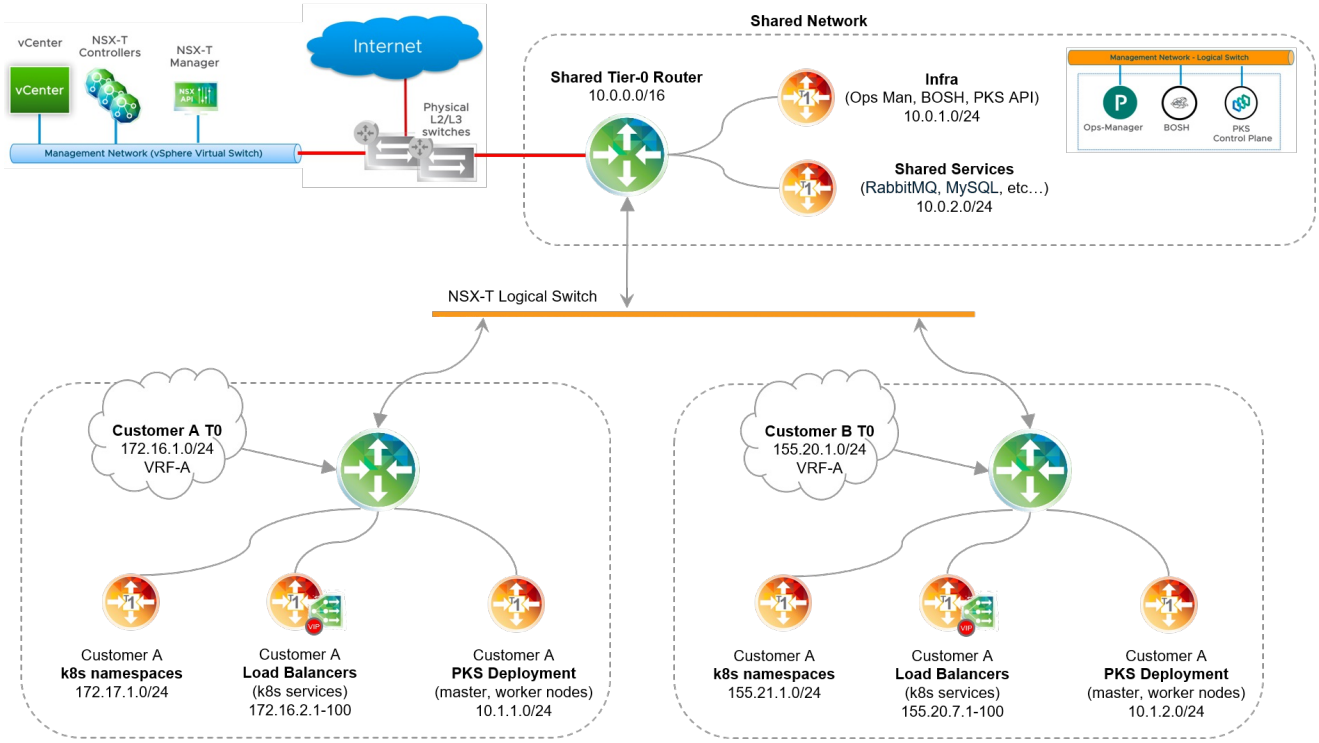
```
{
  "name": "np-custom-fip",
  "description": "Network Profile for Custom Floating IP Pool",
  "parameters": {
    "fip_pool_ids": [
      "e50e8f6e-1a7a-45dc-ad49-3a607baa7fa0",
      "ebe78a74-a5d5-4dde-ba76-9cf4067eee55"
    ]
  }
}
```

The example above uses two floating IP pools. With this configuration, if the first pool of IP addresses, `e50e8f6e-1a7a-45dc-ad49-3a607baa7fa0`, is exhausted, the system will use the IP addresses in the next IP pool that is listed, `ebe78a74-a5d5-4dde-ba76-9cf4067eee55`.

Edge Router Selection

Using PKS on vSphere with NSX-T, you can deploy Kubernetes clusters on dedicated Tier-0 routers, creating a multi-tenant environment for each Kubernetes cluster. As shown in the diagram below, with this configuration a shared Tier-0 router hosts the PKS control plane and connects to each

customer Tier-0 router using BGP. To support multi-tenancy, configure firewall rules and security settings in NSX Manager.



To deploy Kubernetes clusters on tenancy-based Tier-0 router(s), follow the steps below:

1. For each Kubernetes tenant, create a dedicated Tier-0 router, and configure static routes, BGP, NAT and Edge Firewall security rules as required by each tenant. For instructions, see Configuring Multiple Tier-0 Routers for Tenant Isolation.

2. Define a network profile per tenant that references the Tier-0 router UUID provisioned for that tenant. For example, the following network profiles define two tenant Tier-0 routers with a NATed topology.

```
np_customer_A-NAT.json
{
 "description": "network profile for Customer A",
 "name": "network-profile-Customer-A",
 "parameters": {
  "lb_size": "medium",
  "t0_router_id": "82e766f7-67f1-45b2-8023-30e2725600ba",
  "fip_pool_ids": ["8ec655f-009a-79b7-ac22-40d37598c0ff"],
  "pod_ip_block_ids": ["fce766f7-aaf1-49b2-d023-90e272e600ba"]
 }
}
```

```
np_customer_B-NAT.json
{
 "description": "network profile for Customer B",
 "name": "network-profile-Customer-B",
 "parameters": {
  "lb_size": "small",
  "t0_router_id": "a4e766cc-87ff-15bd-9052-a0e2425612b7",
  "fip_pool_ids": ["4ec625f-b09b-29b4-dc24-10d37598c0d1"],
  "pod_ip_block_ids": ["91e7a3a1-c5f1-4912-d023-90e272260090"]
 }
}
```

The following network profiles define two customer Tier-0 routers for a no-NAT topology:

```
np_customer_A.json
{
 "description": "network profile for Customer A",
 "name": "network-profile-Customer-A",
 "parameters": {
  "lb_size": "medium",
  "t0_router_id": "82e766f7-67f1-45b2-8023-30e2725600ba",
  "fip_pool_ids": [
     "8ec655f-009a-79b7-ac22-40d37598c0ff",
     "7ec625f-b09b-29b4-dc24-10d37598c0e0"
  ],
  "pod_routable": true,
  "pod_ip_block_ids": [
     "fce766f7-aaf1-49b2-d023-90e272e600ba",
     "6faf46fd-ccce-4332-92d2-d918adcccce0"
  ]
 }
}
```

```
np_customer_B.json
{
 "description": "network profile for Customer B",
 "name": "network-profile-Customer-B",
 "parameters": {
  "lb_size": "small",
  "t0_router_id": "a4e766cc-87ff-15bd-9052-a0e2425612b7",
  "fip_pool_ids": [
     "4ec625f-b09b-29b4-dc24-10d37598c0d1",
     "6ec625f-b09b-29b4-dc24-10d37598dDd1"
  ],
  "pod_routable": true,
  "pod_ip_block_ids": [
     "91e7a3a1-c5f1-4912-d023-90e272260090",
     "6faf46fd-ccce-4332-92d2-d918adcccce0"
  ]
 }
}
```

Note: The `pod_routable` parameter controls the routing behavior of a tenant Tier-0 router. If the parameter is set to `true`, the custom Pods IP Block subnet is routable and NAT is not used. If `pod_routable` is not present or is set to `false`, the custom Pods IP Block is not routable and the tenant Tier-0 is deployed in NAT mode.

DNS Configuration for Kubernetes Clusters

Using a network profile, you can define one or more DNS servers for use with Kubernetes clusters. Elements in the `nodes_dns` field of a network profile override the DNS server that is configured in the Networking section of the PKS with NSX-T tile. For more information, see Networking.

The `nodes_dns` field accepts an array with up to 3 elements. Each element must be a valid IP address of a DNS server. If you are deploying PKS in a multi-tenant environment with multiple Tier-0 routers, the first DNS server entered should be a shared DNS server. Subsequent entries are typically specific to the tenant.

The following example network profile, `nodes-dns.json`, demonstrates the configuration of the `nodes_dns` parameter with 3 DNS servers. Each entry is the IP address of a DNS server, with the first entry being a public DNS server.

```
nodes-dns.json
{
   "description": "Overwrite Nodes DNS Entry",
   "name": "nodes_dns_multiple",
   "parameters": {
      "nodes_dns": [
         "8.8.8.8", "192.168.115.1", "192.168.116.1"
      ]
   }
}
```

Please send any feedback you have to pks-feedback@pivotal.io.

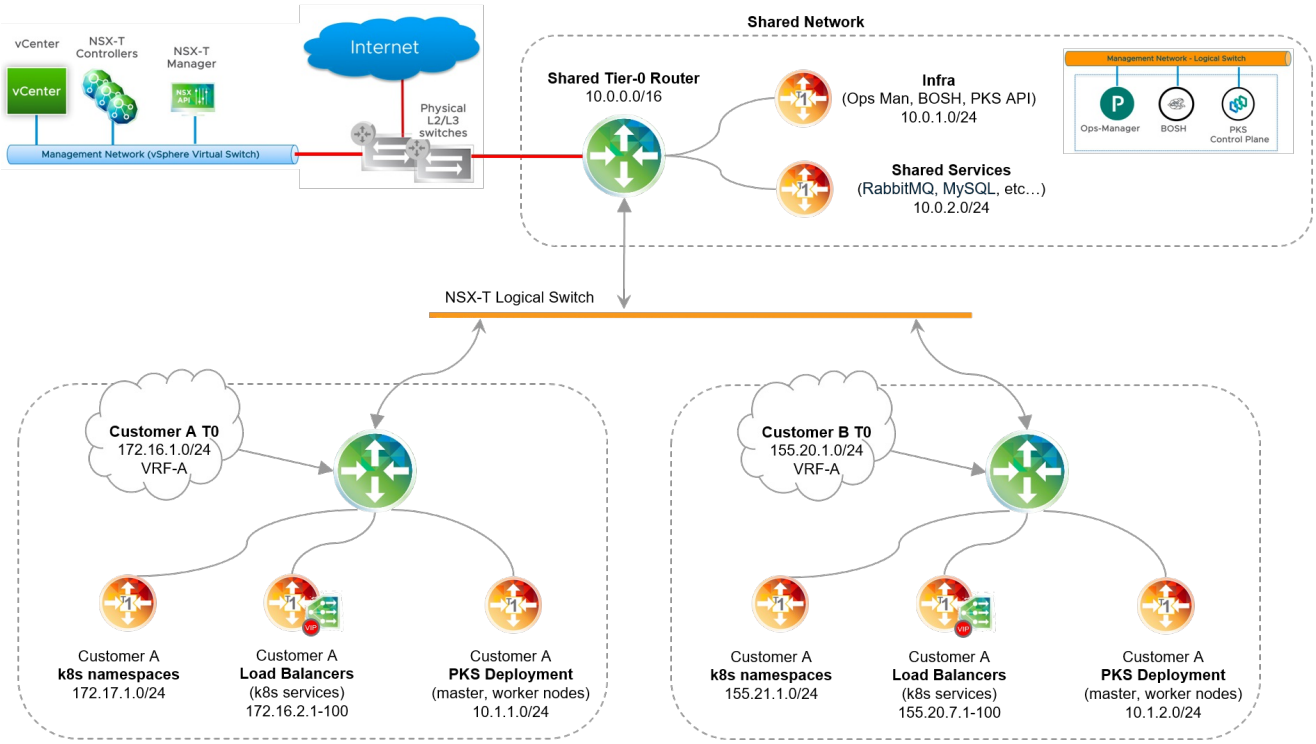
Configuring Multiple Tier-0 Routers for Tenant Isolation

Page last updated:

This topic describes how to create multiple NSX-T Tier-0 (T0) logical routers for use with PKS multi-tenant environments.

About Multi-T0 Router for Tenant Isolation

PKS multi-T0 lets you provision, manage, and secure Kubernetes cluster deployments on isolated tenant networks. As shown in the diagram below, instead of having a single T0 router, there are multiple T0 routers. The Shared Tier-0 router handles traffic between the PKS management network and the vSphere standard network where vCenter and NSX Manager are deployed. There are two Tenant Tier-0 routers that connect to the Shared Tier-0 over an NSX-T logical switch using a VLAN or Overlay transport zone. Using each dedicated T0, Kubernetes clusters are deployed in complete isolation on each tenant network.



Prerequisites

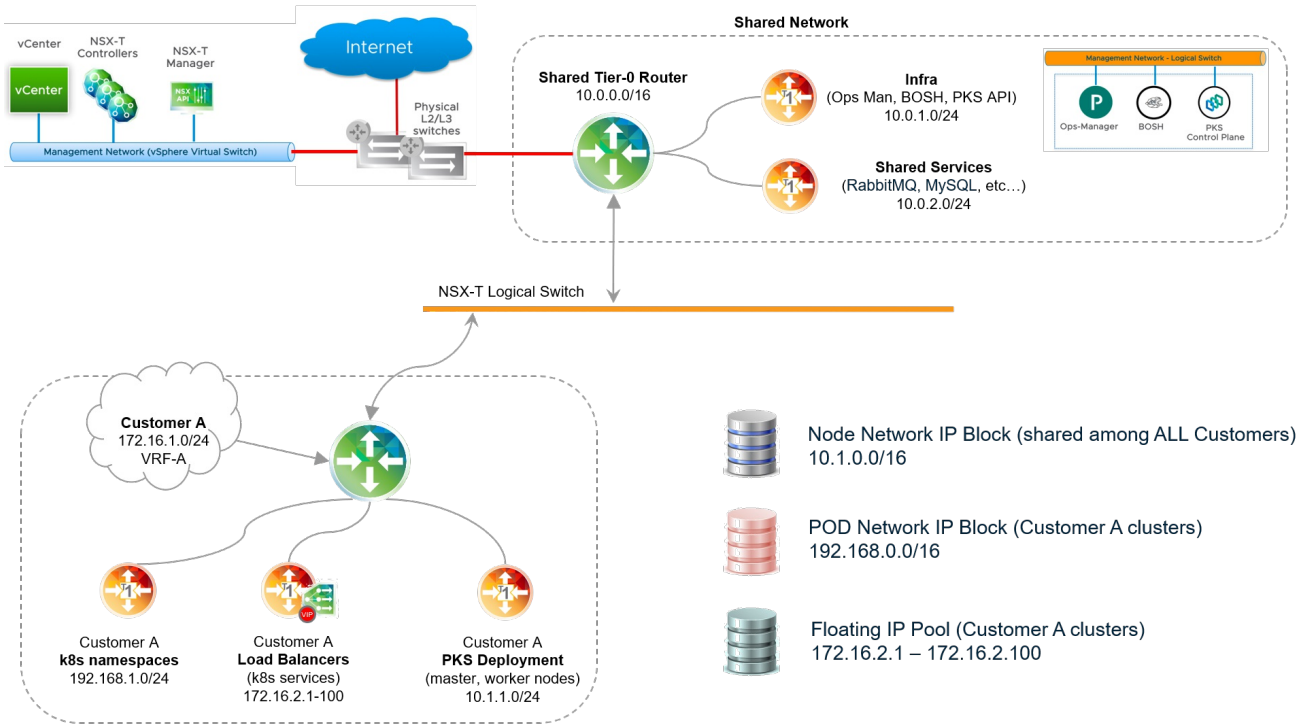
To implement Multi-T0, verify the following prerequisites:

- Supported version of vSphere IaaS is installed. See vSphere with NSX-T Version Requirements.
- Supported version of VMware NSX-T Data Center is installed. See vSphere with NSX-T Version Requirements.
- If you are using NAT mode for the Shared Tier-0 router, review Considerations for NAT Topology on Shared Tier-0 and Considerations for NAT Topology on Tenant Tier-0 before proceeding.

Base Configuration

Step 1: Plan and Provision Additional NSX Edge Nodes for Each Multi-T0 Router

Multi-T0 requires a minimum of four NSX Edge Nodes: Two nodes per T0 operating in active-standby mode. Use the T0 attached to the PKS management plane as the Shared Tier-0 router that connects all T0 routers. In addition, deploy an additional T0 router for each tenant you want to isolate.



Each Tenant Tier-0 router requires a minimum of two NSX Edge Nodes. The formula for determining the minimum number of nodes for all tenants is as follows:

```
2 + (TENANTS x 2)
```

Where `TENANTS` is the number of tenants you want to isolate.

For example, if you want to isolate three tenants, use the following calculation:

```
2 + (3 x 2) = 8 NSX Edge Nodes
```

To isolate ten tenants, use the following calculation:

```
2 + (10 x 2) = 22 NSX Edge Nodes
```

Using the NSX Manager interface, deploy at least the minimum number of Edge Nodes you need for each Tenant Tier-0 and join these Edge Nodes to an Edge Cluster. For more information, see Deploying NSX-T for PKS.

Note: An Edge Cluster can have a maximum of 10 Edge Nodes. If the provisioning requires more Edge Nodes than what a single Edge Cluster can support, multiple Edge Clusters must be deployed.

Step 2: Configure Inter-T0 Logical Switch

All NSX-T Edge Nodes must be connected by a dedicated network provisioned on the physical infrastructure. This network is used to transport traffic across the T0 routers. Plan to allocate a network of sufficient size to accommodate all Tier-0 router interfaces that need to be connected to such network. You must allocate each T0 router one or more IP addresses from that range.

For example, if you plan to deploy two Tenant Tier-0 routers, a subnet with prefix size /28 may be sufficient, such as `50.0.0.0/28`.

Once you have physically connected the Edge Nodes, define a logical switch to connect the Shared Tier-0 router to the Tenant Tier-0 router or routers.

To define a logical switch based on an Overlay or VLAN transport zone, follow the steps below:

1. In NSX Manager, go to **Networking** > **Switching** > **Switches**.

2. Click **Add** and create a logical switch (LS).
3. Name the switch descriptively, such as `inter-t0-logical-switch`.
4. Connect the logical switch to the transport zone defined when deploying NSX-T. For more information, see Deploying NSX-T for PKS.

Step 3: Configure a New Uplink Interface on the Shared Tier-0 Router

The Shared Tier-0 router already has a uplink interface to the external (physical) network that was configured when it was created. For more information, see Create T0 Logical Router.

To enable Multi-T0, you must configure a second uplink interface on the Shared Tier-0 router that connects to the inter-T0 network (`inter-t0-logical-switch` , for example). To do this, complete the following steps:

1. In NSX Manager, go to **Networking > Routers**.
2. Select the Shared Tier-0 router.
3. Select **Configuration** > **Router Ports** and click **Add**.
4. Configure the router port as follows:
    a. For the logical switch, select the inter-T0 logical switch you created in the previous step (for example, `inter-t0-logical-switch`).
    b. Provide an IP address from the allocated range. For example, `50.0.0.1/24` .

Step 4: Provision Tier-0 Router for Each Tenant

Create a Tier-0 logical router for each tenant you want to isolate. For more information, see Tier-0 Logical Router in the NSX-T documentation.

For instructions, see Create T0 Router. When creating each Tenant Tier-0 router, make sure you set the router to be active/passive, and be sure to name the logical switch descriptively, such as `t0-router-customer-A` .

Step 5: Create Two Uplink Interfaces on Each Tenant Tier-0 Router

Similar to the Shared Tier-0 router, each Tenant Tier-0 router requires at a minimum two uplink interfaces.

- The first uplink interface provides an uplink connection from the Tenant Tier-0 router to the tenant's corporate network.
- The second uplink interface provides an uplink connection to the Inter-T0 logical switch that you configured. For example, `inter-t0-logical-switch` .

For instructions, see Create T0 Router. When creating the uplink interface that provides an uplink connection to the Inter-T0 logical switch, be sure to give this uplink interface an IP address from the allocated pool of IP addresses.

Step 6: Verify the Status of the Shared and Tenant Tier-0 Routers

When you have completed the configuration of the Shared and Tenant Tier-0 routers as described above, verify your progress up to this point. On the Shared Tier-0 router, you should have two uplink interfaces, one to the external network and the other to the inter-T0 logical switch. On the Tenant Tier-0 router, you should have two uplink interfaces, one to the inter-T0 logical switch and the other to the external network. Each uplink interface is connected to a transport node.

The images below provide an example checkpoint for verifying the uplink interfaces for the Shared and Tenant Tier-0 routers. In this example, the Shared Tier-0 has one uplink interface at `10.40.206.10/25` on the transport Edge Node `edge-TN1` , and the second uplink interface at `110.40.206.9/25` on the transport Edge Node `edge-TN2` .

☐	Uplink-2	585e.....	Uplink	10.40.206.9/25	uplink-LS1 (8f0831de-01f1...	edge-TN2
☐	Uplink1	e1f5...e...	Uplink	10.40.206.10/25	uplink-LS1 (uplink1-port)	edge-TN1

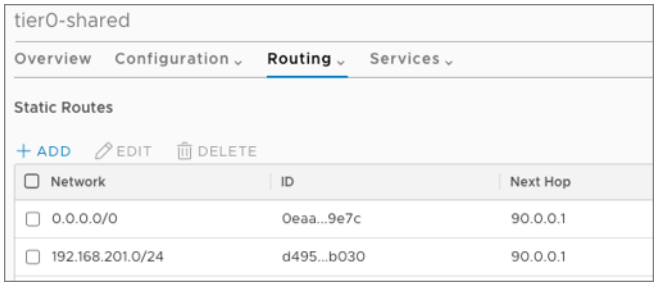
Similarly, the Tenant Tier-0 has one uplink interface at `10.40.206.13/25` on the transport Edge Node `edge-TN3`, and the second uplink interface at `10.40.206.14/25` on the transport Edge Node `edge-TN4`.

Logical Rol	ID	Type	IP Address/mask	Connected To	Transport Node	Relay Service	Statistics
T0-2-u...	4238.....	Uplink	10.40.206.13/25	uplink-LS1 (311a54cb-48d...	edge-TN3		
T0-2-u...	8f15...f...	Uplink	10.40.206.14/24	uplink-LS1 (974cbf11-0b3...	edge-TN4		

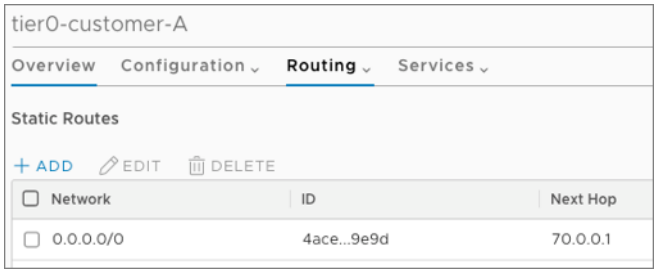
Step 7: Configure Static Routes

For each T0 router, including the Shared Tier-0 and all Tenant Tier-0 routers, define a static route to the external network. For instructions, see Configure a Static Route in the NSX-T documentation.

For the Shared Tier-0 router, the default static route points to the external management components such as vCenter and NSX Manager and provides internet connectivity. As shown in the image below, the Shared Tier-0 defines a static route for vCenter and NSX Manager as `192.168.201.0/24`, and the static route for internet connectivity as `0.0.0.0/0`:



For each Tenant Tier-0 router, the default static route should point to the tenant's corporate network. As shown in the image below, the Tenant Tier-0 defines a static route to the corporate network as `0.0.0.0/0`:



Step 8: Considerations for NAT Topology on Shared Tier-0

The Multi-T0 configuration steps documented here apply to deployments where NAT mode is **not** used on the Shared Tier-0 router. For more information, see NSX-T Deployment Topologies for PKS.

For deployments where NAT-mode is used on the Shared Tier-0 router, additional provisioning steps must be followed to preserve NAT functionality to external networks while bypassing NAT rules for traffic flowing from the Shared Tier-0 router to each Tenant Tier-0 router.

Existing PKS deployments where NAT mode is configured on the Shared Tier-0 router cannot be repurposed to support a Multi-T0 deployment following this documentation.

Step 9: Considerations for NAT Topology on Tenant Tier-0

Note: This step only applies to NAT topologies on the Tenant Tier-0 router. For more information on NAT mode, see NSX-T Deployment Topologies for PKS.

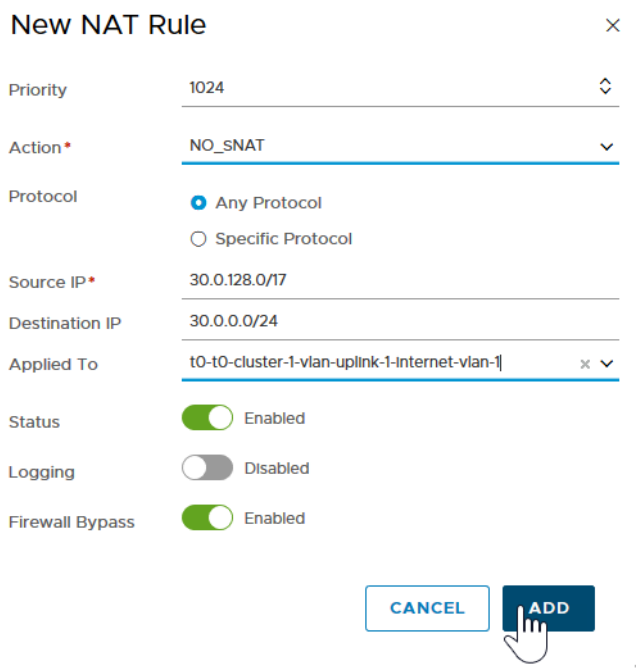
Note: NAT mode for Tenant Tier-0 routers is enabled by defining a non-routable custom Pods IP Block using a Network Profile. For more information, see Defining Network Profiles.

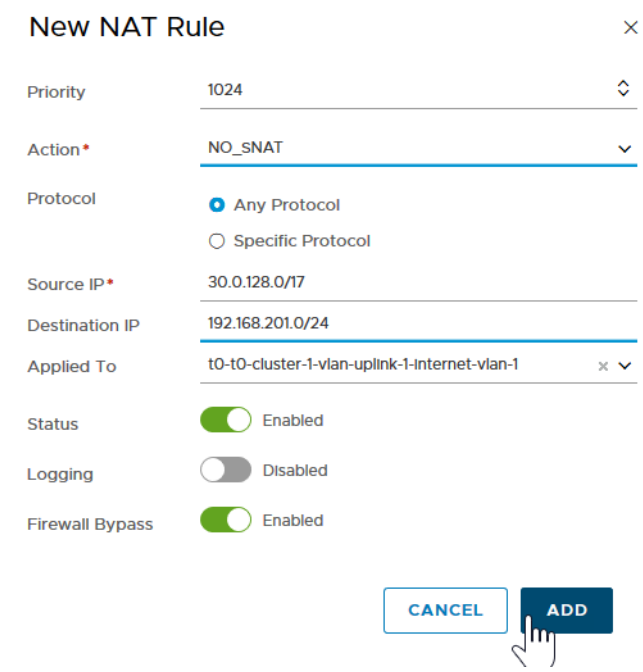
In a Multi-T0 environment with NAT mode, traffic on the Tenant Tier-0 network going from Kubernetes cluster nodes to PKS management components residing on the Shared Tier-0 router must bypass NAT rules. This is required because PKS-managed components such as BOSH Director connect to Kubernetes nodes based on routable connectivity without NAT.

To avoid NAT rules being applied to this class of traffic, you need to create two high-priority **NO_SNAT** rules on each Tenant Tier-0 router. These NO_SNAT rules allow "selective" bypass of NAT for the relevant class of traffic, which in this case is connectivity from Kubernetes node networks to PKS management components such as the PKS API, Ops Manager, and BOSH Director, as well as to infrastructure components such as vCenter and NSX Manager.

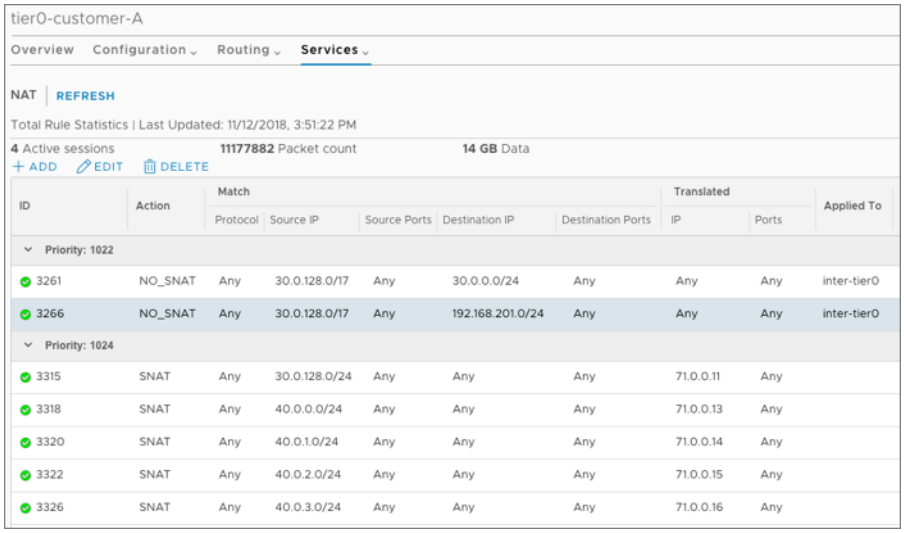
For each Tenant Tier-0 router, define two NO_SNAT rules to classify traffic. The source for both rules is the Nodes IP Block CIDR. The destination for one rule is the PKS Management network where PKS, Ops Manager, and BOSH Director are deployed. The destination for the other rule is the external network where NSX Manager and vCenter are deployed.

For example, the following image shows two NO_SNAT rules created on a Tenant Tier-0 router. The first rule un-NATs traffic from Kubernetes nodes (`30.0.128.0/17`) to the PKS management network (`30.0.0.0/24`). The second rule un-NATs traffic from Kubernetes nodes (`30.0.128.0/17`) to the external network (`192.168.201.0/24`).





The end result is two NO_SNAT rules on each Tenant Tier-0 router that bypass the NAT rules for the specified traffic.



ID	Action	Match					Translated		Applied To
		Protocol	Source IP	Source Ports	Destination IP	Destination Ports	IP	Ports	
Priority: 1022									
3261	NO_SNAT	Any	30.0.128.0/17	Any	30.0.0.0/24	Any	Any	Any	inter-tier0
3266	NO_SNAT	Any	30.0.128.0/17	Any	192.168.201.0/24	Any	Any	Any	inter-tier0
Priority: 1024									
3315	SNAT	Any	30.0.128.0/24	Any	Any	Any	71.0.0.11	Any	
3318	SNAT	Any	40.0.0.0/24	Any	Any	Any	71.0.0.13	Any	
3320	SNAT	Any	40.0.1.0/24	Any	Any	Any	71.0.0.14	Any	
3322	SNAT	Any	40.0.2.0/24	Any	Any	Any	71.0.0.15	Any	
3326	SNAT	Any	40.0.3.0/24	Any	Any	Any	71.0.0.16	Any	

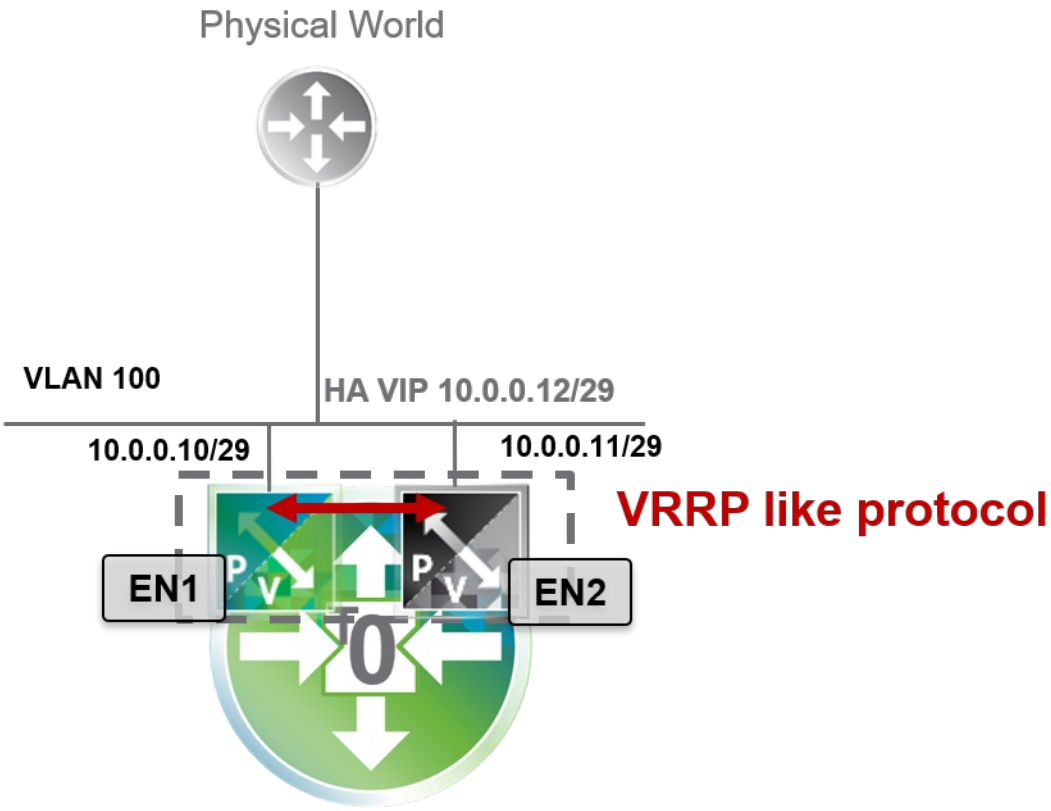
Step 10: Configure BGP on Each Tenant Tier-0 Router

The Border Gateway Protocol (BGP) is used for route redistribution and filtering across all Tier-0 routers. BGP allows the Shared Tier-0 router to dynamically discover the location of Kubernetes clusters (Node networks) deployed on each Tenant Tier-0 router.

In a Multi-T0 deployment, all Tier-0 routers are deployed in Active/Standby mode. As such, special consideration must be given to the network design to preserve reliability and fault tolerance of the Shared and Tenant Tier-0 routers.

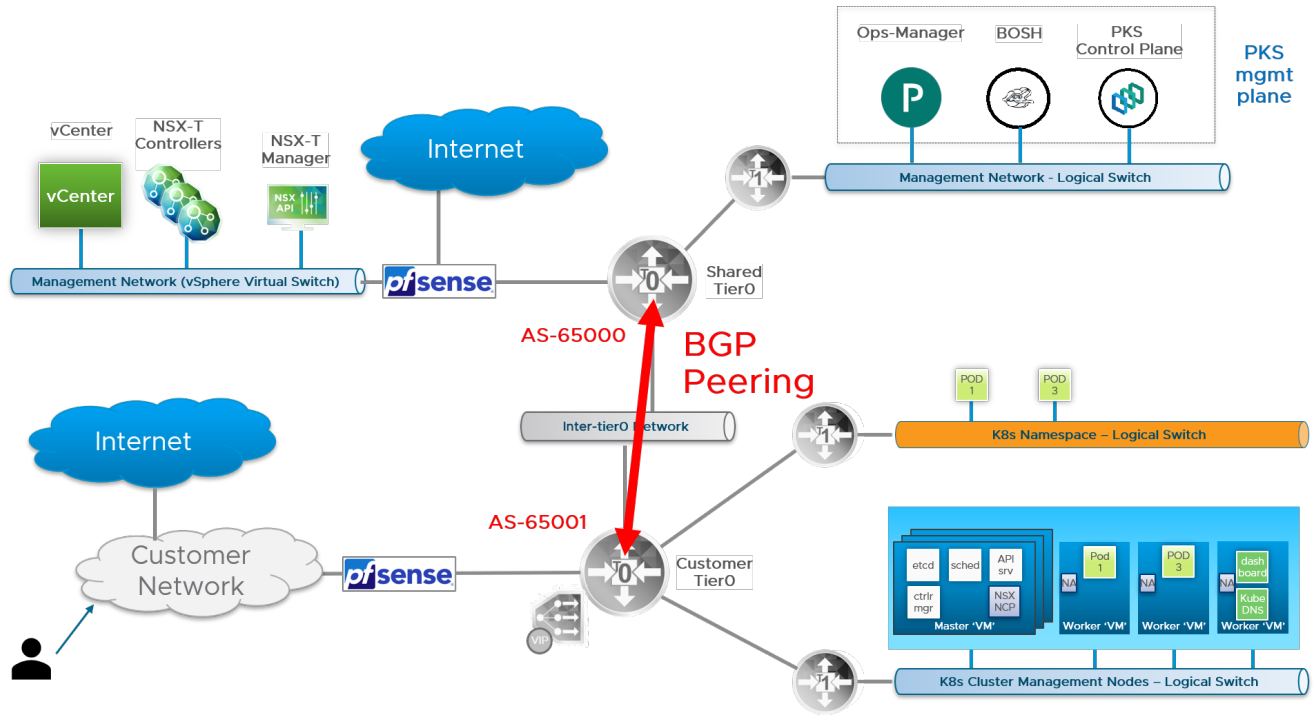
Failover of a logical router is triggered when the router is losing all of its BGP sessions. If multiple BGP sessions are established across different uplink interfaces of a Tier-0 router, failover will only occur if **all** such sessions are lost. Thus, to ensure high availability on the Shared and Tenant Tier-0 routers, BGP can only be configured on uplink interfaces facing the Inter-Tier-0 network. This configuration is shown in the diagram below.

Note: In a Multi-T0 deployment, BGP cannot be configured on external uplink interfaces. Uplink external connectivity must use VIP-HA with NSX-T to provide high availability for external interfaces. For more information, see Configure Edge Nodes for HA.



You must configure BGP routing on each Tier-0 router. The steps that follow are for each Tenant Tier-0 router. The instructions for the Shared Tier-0 are provided in subsequent steps. As a prerequisite, assign a unique Autonomous System Number to each Tier-0 router. Each AS number you assign must be private within the range `64512-65534`. For more information, see Configure BGP on a Tier-0 Logical Router in the NSX-T documentation.

Note: To configure BGP for the Tenant Tier-0, you will need to use the Shared Tier-0 AS number. As such, identify the AS numbers you will use for the Tenant and Shared Tier-0 routers before proceeding.



Configure BGP AS Number

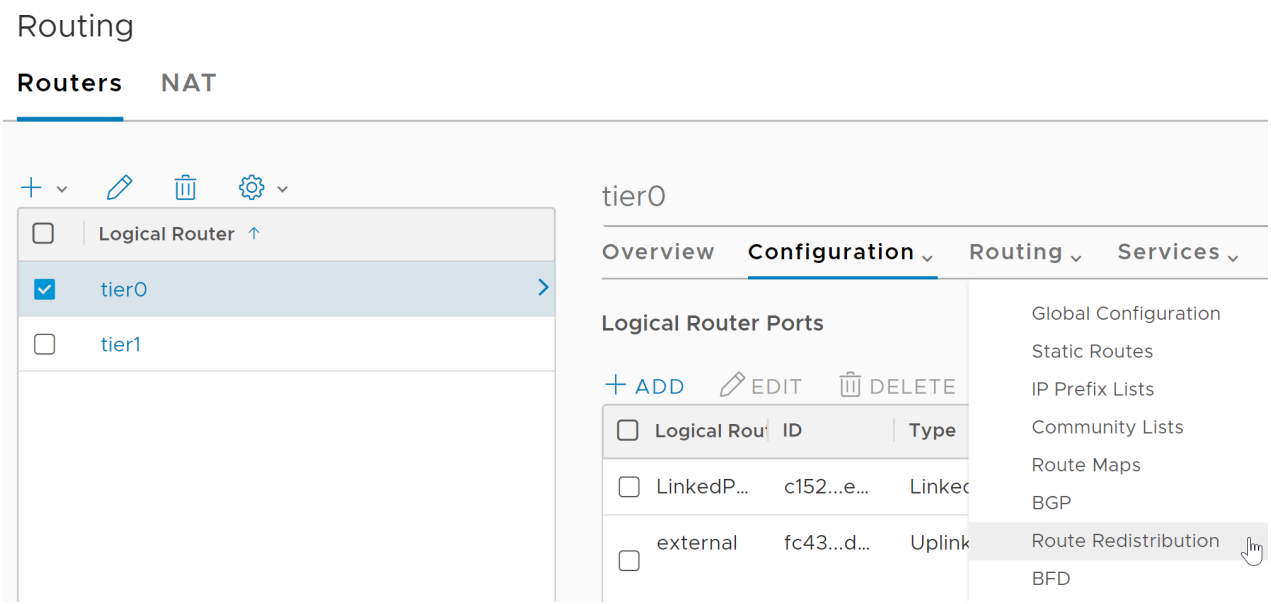
Once you have chosen the AS number for the Tenant Tier-0 router, configure BGP with the chosen AS number as follows:

1. In NSX Manager, select **Networking** > **Routers**.
2. Select the Tenant Tier-0 router.
3. Select **Routing** > **BGP**, the click **ADD**.
4. Add the AS number to the BGP configuration in the `local AS` field.
5. Click on the `enabled` slider to activate BGP.
6. Lastly, disable the ECMP slider.

Configure BGP Route Distribution

To configure BGP route distribution for each Tenant Tier-0 router, follow the steps below:

1. In NSX Manager, select the Tenant Tier-0 router.
2. Select **Routing** > **Route Redistribution**.



3. Click **Add** and configure as follows:
    a. **Name**: NSX Static Route Redistribution
    b. **Sources**: Select **Static**, **NSX Static**, and **NSX Connected**

Configure IP Prefix Lists

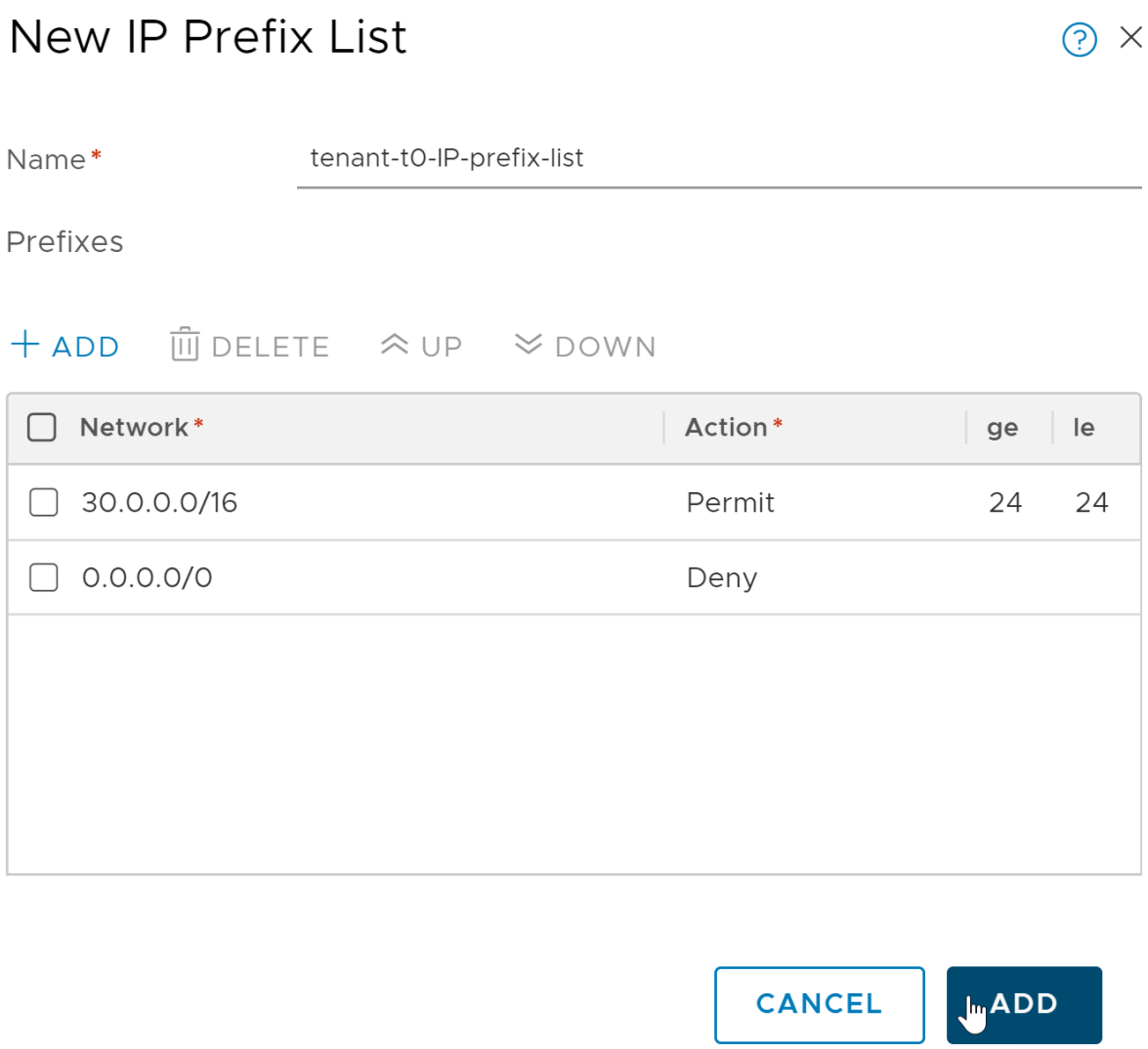
In this step you define an **IP Prefix List** for each Tenant Tier-0 router to advertise any Kubernetes node network of standard prefix size /24, as specified by the less-than-or-equal-to (le) and greater-than-or-equal-to (ge) modifiers in the configuration. The CIDR range to use for the definition of the list entry is represented by the Nodes IP Block network, for example `30.0.0.0/16`.

For more information about IP Prefix Lists, see Create an IP Prefix List in the NSX-T documentation.

To configure an IP Prefix List for each Tenant Tier-0 router, follow the steps below:

1. In NSX Manager, select the Tenant Tier-0 router.
2. Select **Routing** > **IP Prefix Lists**.
3. Click **Add** and configure as follows:
    a. **Name**: Enter a descriptive name.
    b. Click **Add** and create a **Permit** rule that allows redistribution of the exact /24 network, carved from the **Nodes IP Block**.

c. Click **Add** and create a **Deny** rule that denies everything else on the network `0.0.0.0/0`.



Configure BGP Peer

To configure BGP peering for each Tenant Tier-0 router, follow the steps below:

1. In NSX Manager, select the Tenant Tier-0 router.
2. Go to **Routing** > **BGP**.
3. Click **Add** and configure the BGP rule as follows:
    a. **Neighbor Address**: Enter the IP address of the Shared Tier-0 router.
    b. **Local Address**: Select the individual uplink interfaces facing the inter-tier0 logical switch.
    c. **Address Families**: Click **Add** and configure as follows:
        i. **Type**: IPV4_UNICAST
        ii. **State**: Enabled
        iii. **Out Filter**: Select the IP Prefix List created above.
        iv. Click **Add**.
    d. Back at the **Routing** > **BGP** screen:
        i. Enter the Shared Tier-0 AS number.
        ii. After creating the BGP neighbor, select **Edit** and click **Enable BGP**.

Step 11: Configure BGP on the Shared Tier-0 Router

The configuration of BGP on the Shared Tier-0 is similar to the BGP configuration each Tenant Tier-0, with the exception of the IP Prefix list that permits traffic to the PKS management network where PKS, BOSH, and Ops Manager are located.

As with each Tenant Tier-0 router, you will need to assign a unique private AS number within the private range `64512-65534` to the Shared Tier-0 router. Once the AS number is assigned, use NSX Manager to configure the following BGP rules for the Shared Tier-0 router.

Configure BGP AS Number

To configure BGP on the Shared Tier-0 with the AS number, complete the corresponding set of instructions in the tenant BGP section above.

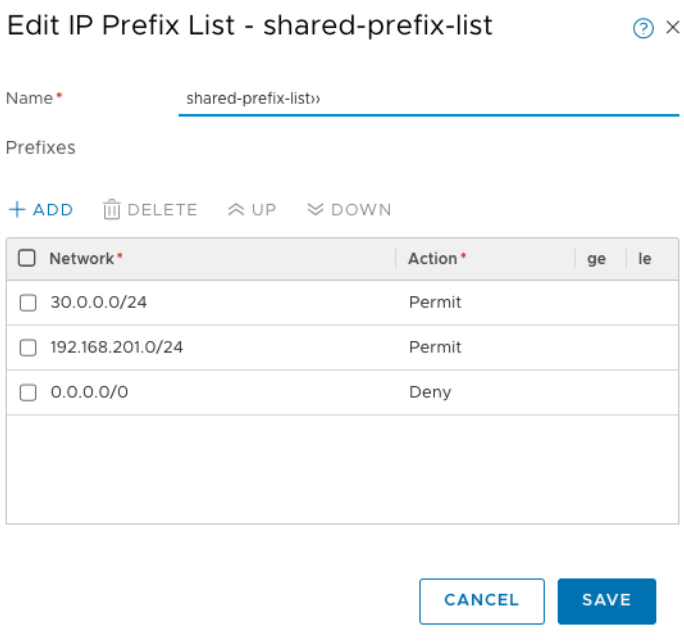
Configure BGP Route Distribution

To configure BGP route distribution for the Shared Tier-0 router, complete the corresponding set of instructions in the BGP tenant section above.

Configure IP Prefix Lists

To configure IP prefix lists for each Tenant Tier-0 router, follow the steps below:

1. In NSX Manager, select the Tenant Tier-0 router.
2. Select **Routing** > **IP Prefix Lists**.
3. Click **Add** and configure as follows:
    a. **Name**: Enter a descriptive name.
    b. Click **Add** and create a **Permit** rule for the infrastructure components vCenter and NSX Manager.
    c. Click **Add** and create a **Permit** rule for the PKS management components (PKS, Ops Manager, and BOSH).
    d. Click **Add** and create a **Deny** rule that denies everything else on the network `0.0.0.0/0`.



Configure BGP Peer

1. In NSX Manager, select the Tenant Tier-0 router.
2. Go to **Routing** > **BGP**.
3. Click **Add** and configure the BGP rule as follows:

a. **Neighbor Address**: Enter the IP address of the Shared Tier-0 router.
b. **Local Address**: Select **All Uplinks**.
c. **Address Families**: Click **Add** and configure as follows:

   i. **Type**: IPV4_UNICAST
   ii. **State**: Enabled
   iii. **Out Filter**: Select the IP Prefix List that includes the network where vCenter and NSX Manager are deployed, as well as the network where the PKS management plane is deployed.
   iv. Click **Add**.

d. Back at the **Routing** > **BGP** screen:

   i. Enter the Tenant Tier-0 AS number.
   ii. After creating the BGP neighbor, select **Edit** and click **Enable BGP**.

Note: You must repeat this step for each Tenant Tier-0 router you want to peer with the Shared Tier-0 router.

Step 12: Test the Base Configuration

Perform the following validation checks for all Tier-0 routers. You should perform the validation checks on the Shared Tier-0 first followed by each Tenant Tier-0 router. For each Tier-0, the validation should alternate among checking for the BGP summary and the router Routing Table.

Shared Tier-0 Validation

Verify that the Shared Tier-0 has an active peer connection to each Tenant Tier-0 router. To verify BGP Peering.

- In NSX Manager, select the Shared Tier-0 router and choose **Actions** > **Generate BGP Summary**.
- Validate that the Shared Tier-0 router has one active peer connection to each Tenant Tier-0 router.

Verify that the Shared Tier-0 routing table includes all BGP routes to each Shared Tier-0.

- In NSX Manager, select **Networking** > **Routers** > **Routing**.
- Select the Shared Tier-0 router and choose **Actions** > **Download Routing Table**.
- Download the routing table for the Shared Tier-0 and verify the routes.

Tenant Tier-0 Validation

Verify that the Shared Tier-0 has an active peer connection to each Tenant Tier-0 router. To verify BGP Peering.

- In NSX Manager, select the Tenant Tier-0 router and choose **Actions** > **Generate BGP Summary**.
- Validate that the Tenant Tier-0 router has one active peer connection to the Shared Tier-0 router.
- Repeat for all other Tenant Tier-0 routers.

Verify that the T0 routing table for each Tenant Tier-0 includes all BGP routes to reach vCenter, NSX Manager, and the PKS management network.

- In NSX Manager, select **Networking** > **Routers** > **Routing**.
- Select the T0 router and choose **Actions** > **Download Routing Table**.
- Download the routing table for each of the Tenant Tier-0 routers.

Note: At this point, the Shared Tier-0 has no BGP routes because you have not deployed any Kubernetes clusters. The Shared Tier-0 will show BGP routes when you deploy Kubernetes clusters to the Tenant Tier-0 routers. Each Tenant Tier-0 router shows a BGP exported route that makes each Tenant Tier-0 router aware of the PKS management network and other external networks where NSX-T and vCenter are deployed.

Security Configuration

In a multi-T0 environment, you can secure two types of traffic:

- Traffic between tenants. See Secure Inter-Tenant Communications.

- Traffic between clusters in the same tenant. See Secure Intra-Tenant Communications.

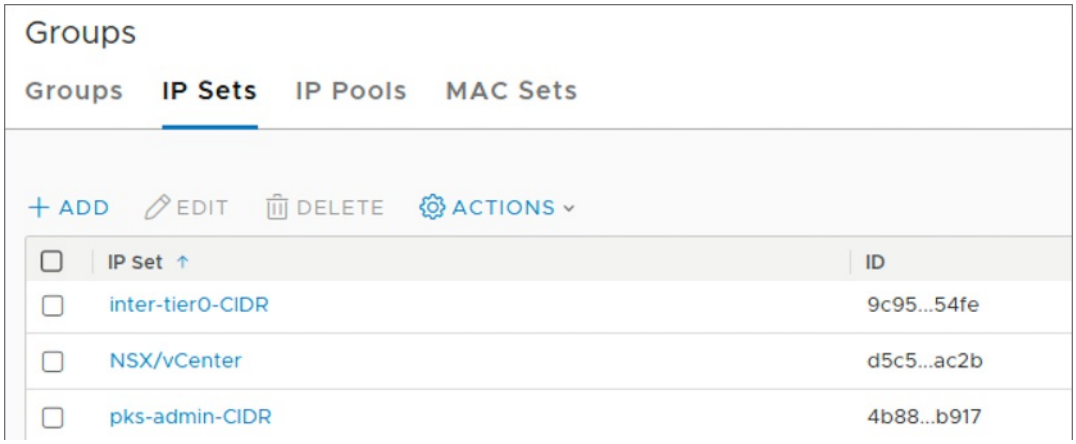
Secure Inter-Tenant Communications

Securing traffic between tenants isolates each tenant and ensures the traffic between the Tenant Tier-0 routers and the Shared Tier-0 router is restricted to the legitimate traffic path.

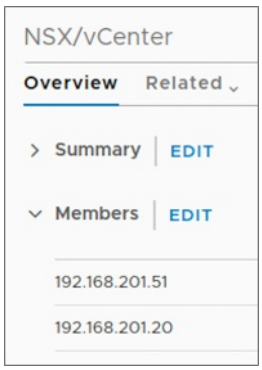
Step 1: Define IP Sets

In NSX-T an **IP Set** is a group of IP addresses that you can use as sources and destinations in firewall rules. For a Multi-T0 deployment you need to create several IP Sets as described below. For more information about creating IP Sets, see Create an IP Set in the NSX-T documentation.

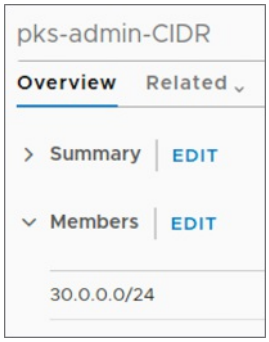
The image below shows a summary of the three required IP Sets you will need to create for securing Multi-T0 deployments:



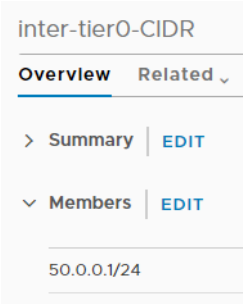
First, define an IP Set that includes the IP addresses for the NSX Manager and vCenter hosts. In the following IP Set example, `192.168.201.51` is the IP address for NSX and `192.168.201.20` is the IP address for vCenter.



Next, define an IP Set that includes the network CIDR for PKS management components. In the following IP Set example, `30.0.0.0/24` is the CIDR block for the PKS Management network.



Lastly, define an IP Set for the the Inter-T0 CIDR created during the base configuration.



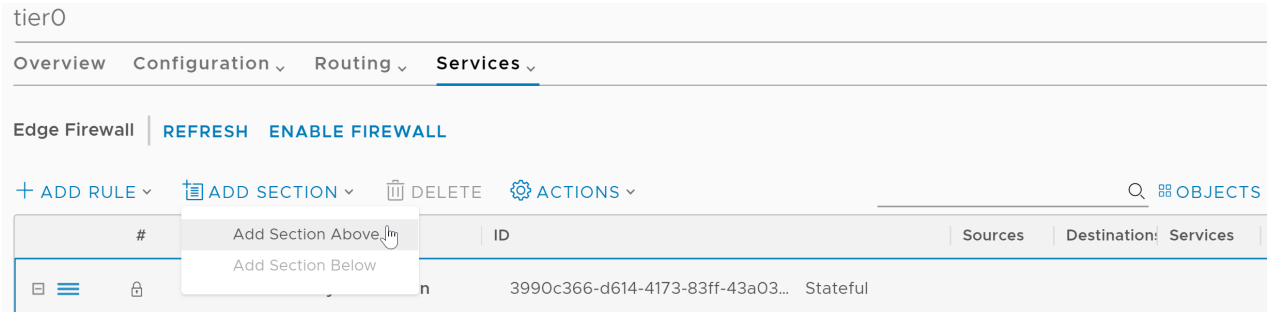
Note: These are the minimum IP Sets you need to create. You may want to define additional IP Sets for convenience.

Step 2: Create Edge Firewall

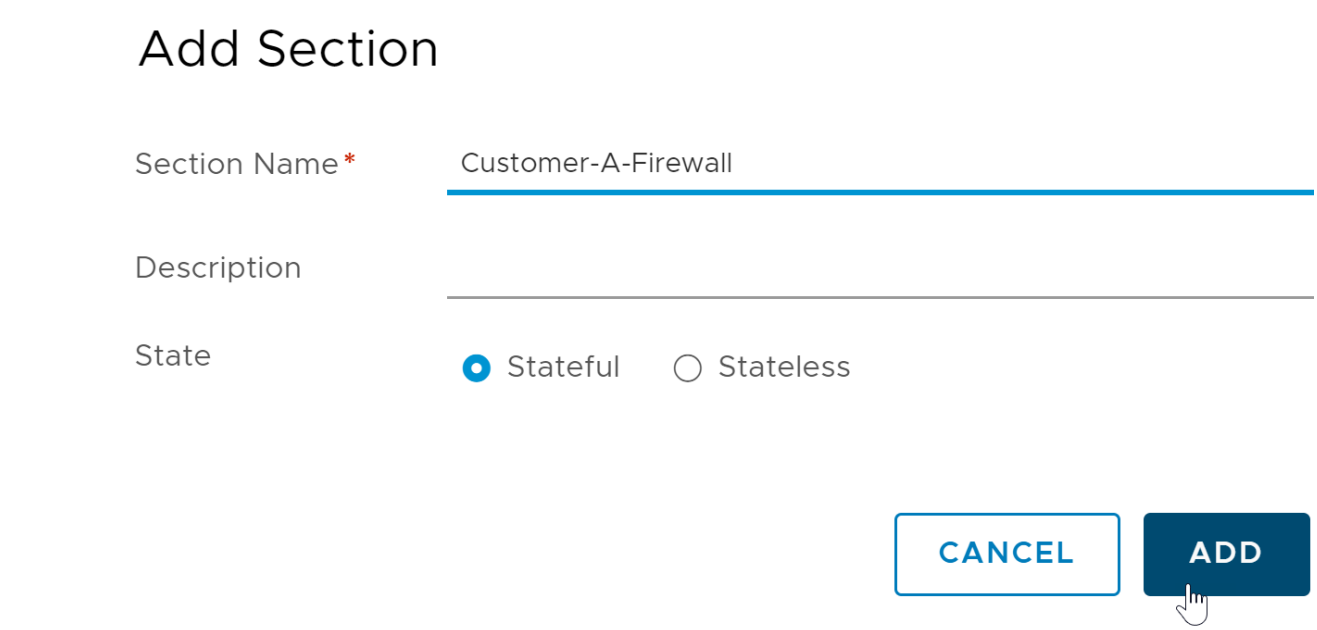
NSX-T Data Center uses Edge Firewall sections and rules to specify traffic handling in and out of the network. A firewall section is a collection of firewall rules. For more information, see About Firewall Rules in the NSX-T documentation.

For each Tenant Tier-0 router, create an Edge Firewall and section as follows:

1. In NSX Manager, go to **Networking > Routers**.
2. Select the Tenant Tier-0 router and click **Services > Edge Firewall**.
3. Select the **Default LR Layer 3 Section**.
4. Click **Add Section** > **Add Section Above**.



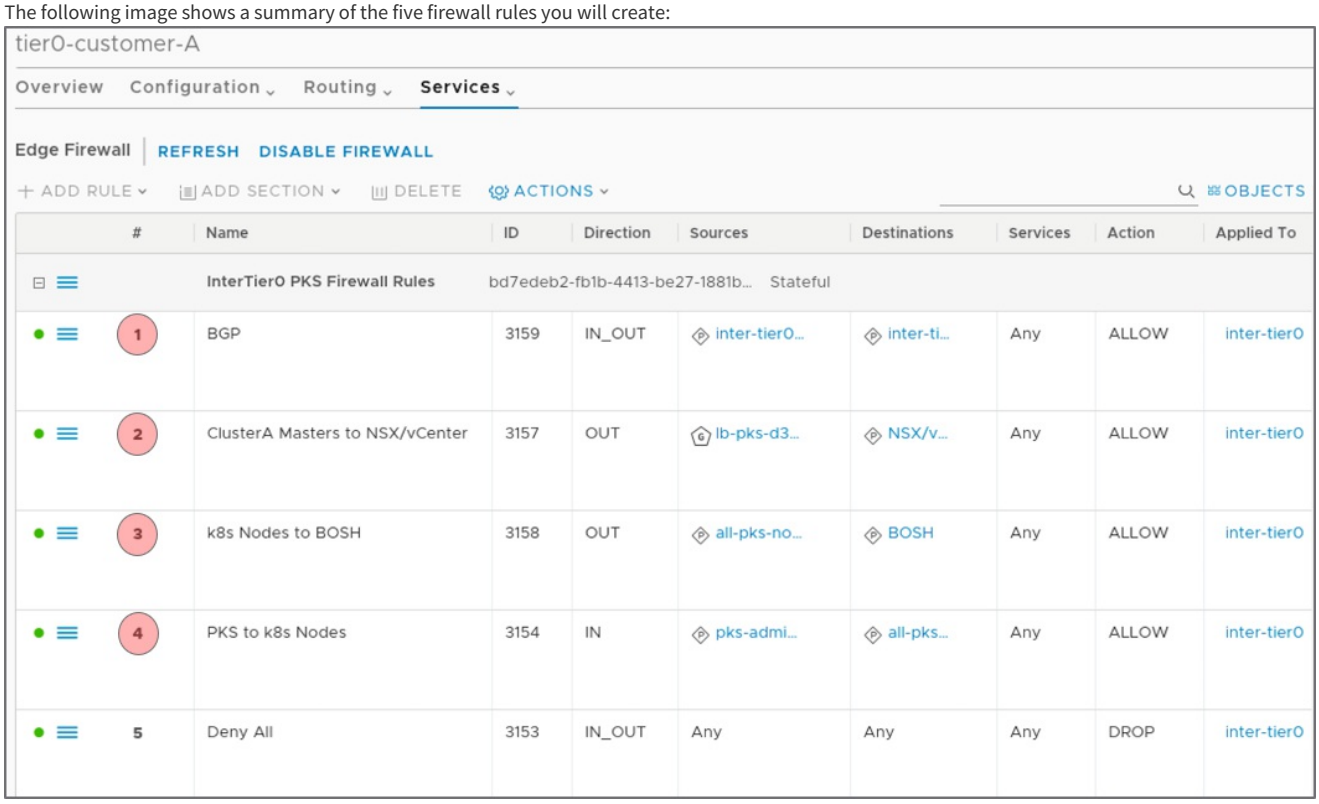
5. Configure the section as follows:
   a. **Section Name**: Enter a unique name for the firewall section.
   b. **State**: **Stateful**



Step 3: Add Firewall Rules

The last step is to define several firewall rules for the Edge Firewall. The firewall rules allow only legitimate control plane traffic to traverse the inter-Tier-0 logical switch, and deny all other traffic.

The following image shows a summary of the five firewall rules you will create:



Note: All firewall rules are applied to the Inter-T0-Uplink interface.

Select the Edge Firewall **Section** you just created, then select **Add Rule**. Add the following five firewall rules:

BGP Firewall Rule

- **Name**: `BGP`
- **Direction**: in and out
- **Source**: IP Set defined for the Inter-T0 CIDR
- **Destination**: IP Set for Inter-T0 CIDR
- **Service**: Any
- **Action**: Allow
- Apply the rule to the Inter-T0-Uplink interface.
- Save the firewall rule.

Clusters Masters Firewall Rule

The source for this firewall rule is a Namespace Group (NSGroup) you define in NSX Manager. The NSGroup is the Bootstrap Security Group specified in the Network Profile associated with this tenant. See Bootstrap Security Group (NSGroup).

Once you have defined the NSGroup, configure the firewall rule as follows.

- **Name**: `Clusters-Masters-to-NSX-and-VC`
- **Direction**: out
- **Source**: NSGroup for Kubernetes Master Nodes
- **Destination**: IP Set for Inter-T0 CIDR
- **Service**: Any
- **Action**: Allow
- Apply the rule to the Inter-T0-Uplink interface.
- Save the firewall rule.

Node Network to Management Firewall Rule

This firewall rule allows Kubernetes node traffic to reach PKS management VMs and the standard network.

- **Name**: `Node-Network-to-Management`
- **Direction**: out
- **Source**: IP Set defined for the Nodes IP Block network
- **Destination**: IP Sets defined for vCenter, NSX Manager, and PKS management plane components
- **Service**: Any
- **Action**: Allow
- Apply the rule to the Inter-T0-Uplink interface.
- Save the firewall rule.

PKS Firewall Rule

This firewall rule allows PKS management plane components to talk to Kubernetes nodes.

- **Name**: `PKS-to-Node-Network`
- **Direction**: ingress
- **Source**: IP Set defined for the PKS management network
- **Destination**: IP Set defined for the Nodes IP Block network
- **Service**: Any
- **Action**: Allow
- Apply the rule to the Inter-T0-Uplink interface.
- Save the firewall rule.

Deny All Firewall Rule

- **Name**: `Deny All`. This setting drops all other traffic that does not meet the criteria of the first three rules.
- **Direction**: in and out
- **Source**: Any
- **Destination**: Any
- **Service**: Any
- **Action**: Drop
- Apply the rule to the Inter-T0-Uplink interface.
- Save the firewall rule.

(Optional) Step 4: Create DFW Section

To use distributed firewall (DWF) rules, you must create a DFW section for the DFW rule set. The DFW section must exist before you create a Kubernetes cluster.

This optional step is recommended for inter-tenant security. It is required for intra-tenant security as described in Secure Intra-Tenant Communications. Because you need to create the DFW section only once, you can use the DFW section you configure in this step when defining DFW rules for intra-tenant communications.

Even if you do not currently plan to use DFW rules, you can create the DFW section and use it later if you decide to define DFW rules. Those rules will apply to any cluster created after you define the DFW section for the tenant Tier-0 router.

Note: You must perform this procedure before you deploy a Kubernetes cluster to the target tenant Tier-0 router.

1. In NSX Manager, navigate to **Security > DFW**, select the top-most rule, and click **Add Section Above**.
2. Configure the section as follows:
    a. In the **Section Name** field, enter a name for your DFW section. For example, `pks-dfw`.
    b. Use the defaults for all other settings on the **New Section** page.
    c. Navigate to the **Manage Tags** page and add a new tag.
        i. In the **Tag** field, enter `top`.
        ii. In the **Scope** field, enter `ncp/fw_sect_marker`.

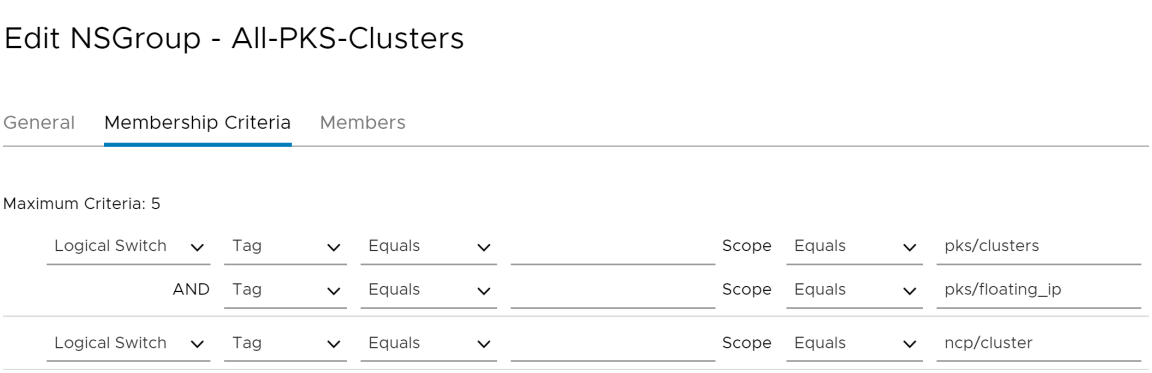
Secure Intra-Tenant Communications

To secure communication between clusters in the same tenancy, you must disallow any form of communication between Kubernetes clusters created by PKS. Securing inter-cluster communications is achieved by provisioning security groups and DFW rules.

Note: You must perform the global procedures, the first three steps described below, before you deploy a Kubernetes cluster to the target tenant Tier-0 router.

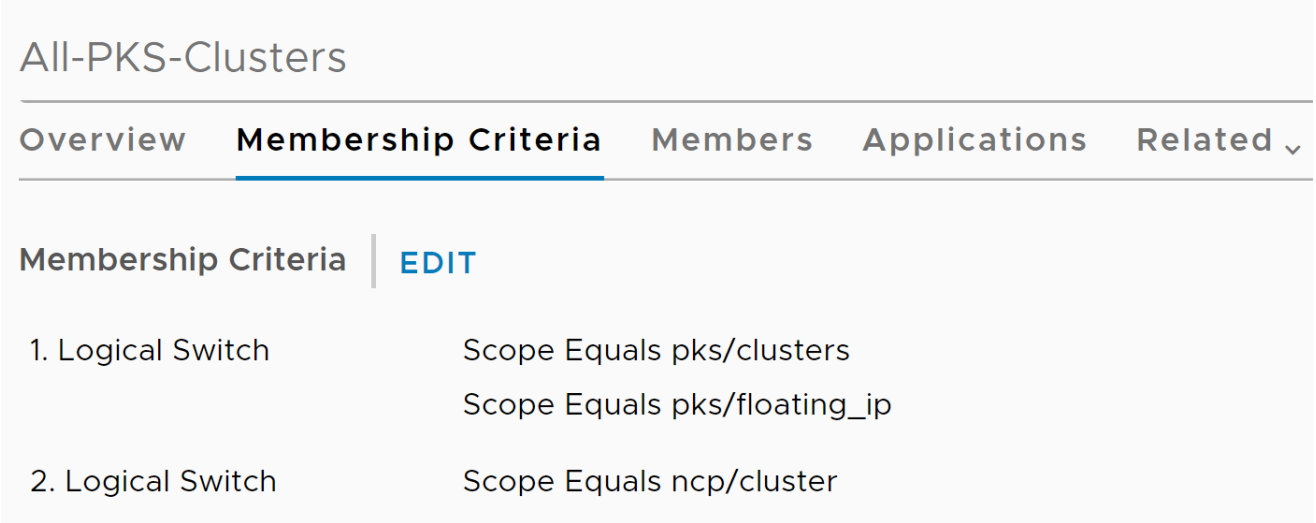
Step 1: Create NS Group for All PKS Clusters

1. In NSX Manager, navigate to **Inventory > Groups > Groups** and **Add new group**.
2. Configure the new NSGroup as follows:
    a. In the **Name** field, enter `All-PKS-Clusters`.
    b. In the **Membership Criteria** tab, add the following two criteria:
        i. For the first criterion, select **Logical switch**.
        ii. For **Scope** > **Equals**, enter `pks/clusters`.
        iii. For **Scope** > **Equals**, enter `pks/floating_ip`.
        iv. For the second criterion, select **Logical switch**.
        v. For **Scope** > **Equals**, enter `ncp/cluster`.



Note: The `pks/clusters`, `pks/floating_ip`, or `ncp/cluster` values are the exact values you must enter when configuring **Scope** > **Equals**. They map to NSX-T objects.

After you configure the `All-PKS-Clusters` NSGroup, the **Membership Criteria** tab looks as follows:



Step 2: Create DFW Section

Before you create distributed firewall rules, you must create a DFW section for the DFW rule set you define later.

To create a DFW section, follow the instructions in Create DFW Section.

Step 3: Create NS Groups

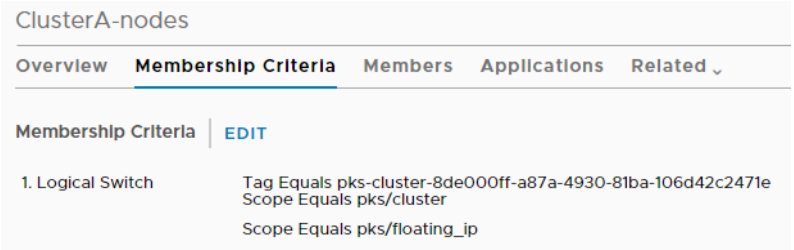
Before creating NS groups, retrieve the UUID of the cluster that you want to secure. To retrieve the cluster UUID, run the `pks cluster YOUR-CLUSTER-NAME` command. For more information about the PKS CLI, see PKS CLI.

Create NS Group for Cluster Nodes

1. In NSX Manager, navigate to **Inventory > Groups > Groups** and click **Add new group**.
2. Configure the new NSGroup as follows:
    a. In the **Name** field, enter the cluster UUID or cluster name and append `-nodes` to the end of the name to distinguish it. The cluster name must be unique.
    b. In the **Membership Criteria** tab, add the following criterion:
        i. Select **Logical Switch**.
        ii. For **Tag** > **Equals**, enter `pks-cluster-YOUR-CLUSTER-UUID`.
        iii. For **Scope** > **Equals**, enter `pks/cluster`.

iv. For **Scope** > **Equals**, enter `pks/floating_ip`. For this scope, leave the **Tag** field empty as shown in the image below.

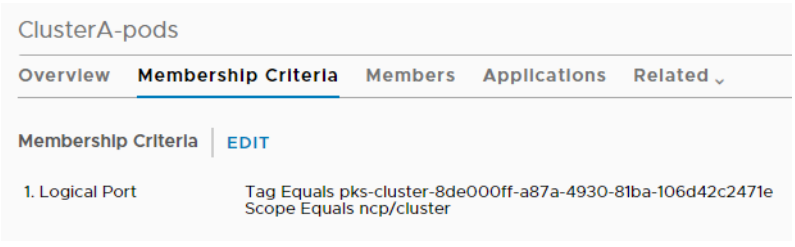
After you configure the NSGroup for cluster nodes, the **Membership Criteria** tab looks as follows:



Create NS Group for Cluster Pods

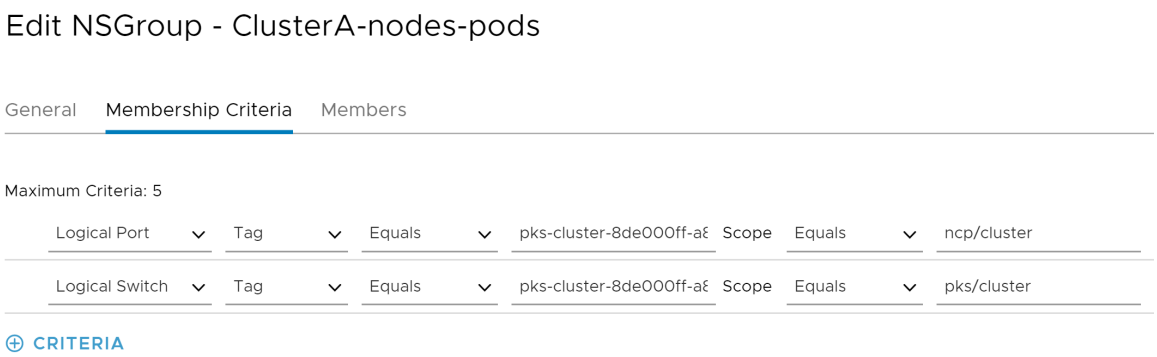
1. In NSX Manager, navigate to **Inventory > Groups > Groups** and click **Add new group**.
2. Configure the new NSGroup as follows:
   a. In the **Name** field, enter the cluster UUID or cluster name and append `-pods` to the end of the name to distinguish it. The cluster name must be unique.
   b. In the **Membership Criteria** tab, add the following criterion:
      i. Select **Logical Port**.
      ii. For **Tag** > **Equals**, enter `pks-cluster-YOUR-CLUSTER-UUID`.
      iii. For **Scope** > **Equals**, enter `ncp/cluster`.

After you configure the NSGroup for cluster pods, the **Membership Criteria** tab looks as follows:

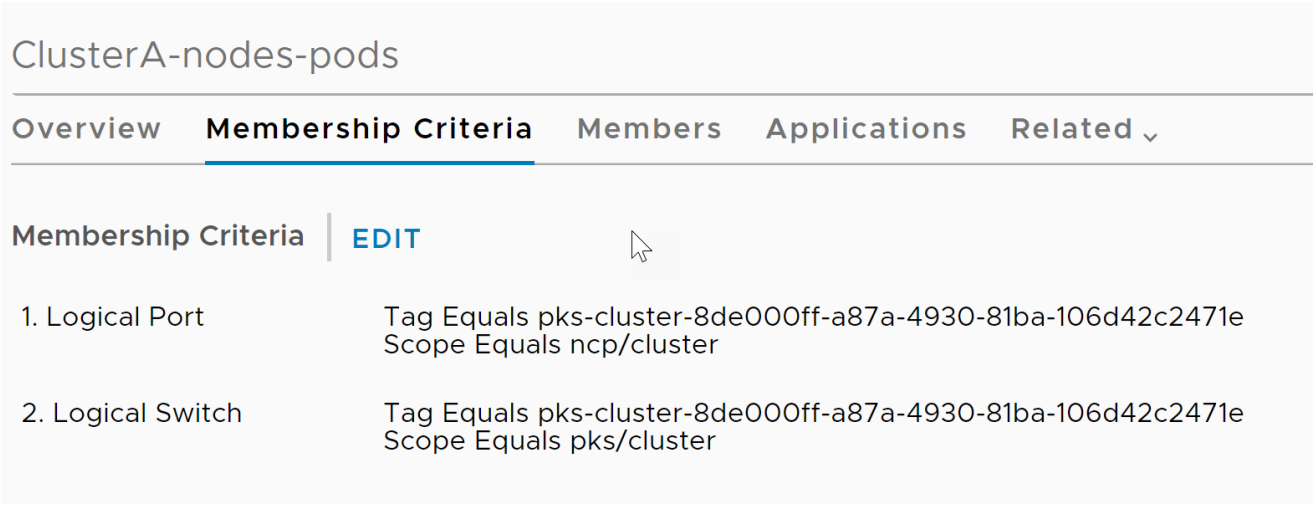


Create NS Group for Cluster Nodes and Pods

1. In NSX Manager, navigate to **Inventory > Groups > Groups** and click **Add new group**.
2. Configure the new NSGroup as follows:
   a. In the **Name** field, enter the cluster UUID or cluster name and append `-nodes-pods` to the end of the name to distinguish it. The cluster name must be unique.
   b. In the **Membership Criteria** tab, add the following two criteria:
      i. For the first criterion, select **Logical Port**.
      ii. For **Tag** > **Equals**, enter `pks-cluster-YOUR-CLUSTER-UUID`.
      iii. For **Scope** > **Equals**, enter `ncp/cluster`.
      iv. For the second criterion, select **Logical Switch**.
      v. For **Tag** > **Equals**, enter `pks-cluster-YOUR-CLUSTER-UUID`.
      vi. For **Scope** > **Equals**, enter `pks/cluster`.



After you configure the NSGroup for cluster nodes and pods, the **Membership Criteria** tab looks as follows:



Step 4: Create DFW Rules

Select the DFW section you created above and configure the following three DFW rules.

DFW Rule 1: Deny Everything Else

This is a global deny rule. Configure the rule as follows:

1. Click **Add Rule**.
2. In the **Name** field, enter a name for your DFW rule.
3. For **Source**, select the `All-PKS-Clusters` NSGroup.

4. For **Destination**, select the `All-PKS-Clusters` NSGroup.
5. For **Service**, select **Any**.
6. For **Apply To**, select the `YOUR-CLUSTER-UUID-nodes-pods` NSGroup.
7. For **Action**, select **Drop**.

DFW Rule 2: Disable Pod to Node Communication

Configure this rule as follows:

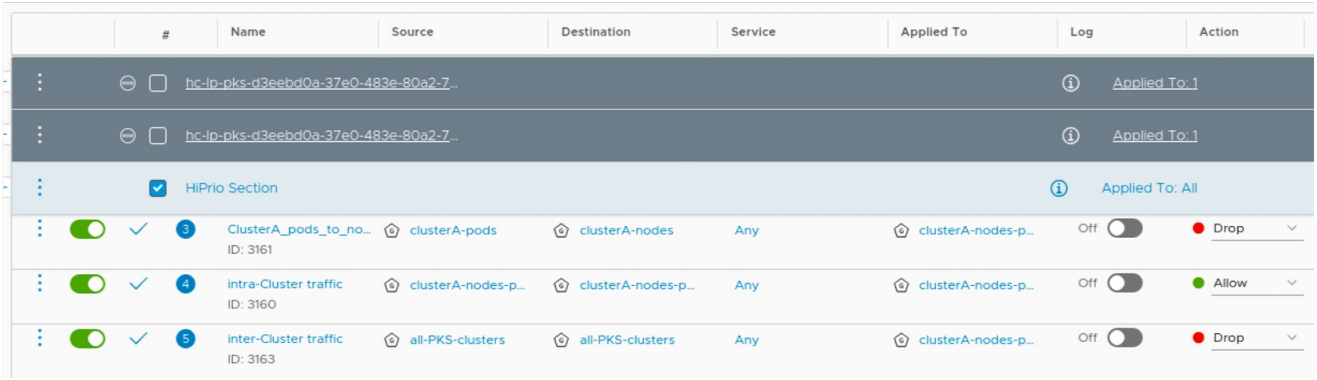
1. Click **Add Rule**.
2. In the **Name** field, enter a name for your DFW rule.
3. For **Source**, select the `YOUR-CLUSTER-UUID-pods` NSGroup.
4. For **Destination**, select `YOUR-CLUSTER-UUID-nodes` NSGroup.
5. For **Service**, select **Any**.
6. For **Apply To**, select the `YOUR-CLUSTER-UUID-nodes-pods` NSGroup.
7. For **Action**, select **Drop**.

DFW Rule 3: Allow Node to Node and Nodes to Pods Communications

Configure this rule as follows:

1. Click **Add Rule**.
2. In the **Name** field, enter a name for your DFW rule.
3. For **Source**, select the `YOUR-CLUSTER-UUID-nodes-pods` NSGroup.
4. For **Destination**, select `YOUR-CLUSTER-UUID-nodes-pods` NSGroup.
5. For **Service**, select **Any**.
6. For **Apply To**, select `YOUR-CLUSTER-UUID-nodes-pods` NSGroup.
7. For **Action**, select **Allow**.

For example, see the three configured DFW rules below:



Please send any feedback you have to pks-feedback@pivotal.io.

Google Cloud Platform (GCP)

This topic lists the steps to follow when installing Pivotal Container Service (PKS) on Google Cloud Platform (GCP).

See the following topics:

- GCP Prerequisites and Resource Requirements
- Installing PKS on GCP

Installing the PKS and Kubernetes CLIs

The PKS and Kubernetes CLIs help you interact with your PKS-provisioned Kubernetes clusters and Kubernetes workloads. To install the CLIs, follow the instructions below:

- Installing the PKS CLI
- Installing the Kubernetes CLI

Please send any feedback you have to pks-feedback@pivotal.io.

GCP Prerequisites and Resource Requirements

Page last updated:

This topic describes the prerequisites and resource requirements for installing Pivotal Container Service (PKS) on Google Cloud Platform (GCP).

Prerequisites

Before you install PKS, you must install either Ops Manager v2.3.1 or later, or Ops Manager v2.4.x. See Install and Configure Ops Manager.

You can install PKS on GCP manually or by using Terraform. If you are installing PKS manually, before you install PKS, you must also do the following:

- Create service accounts for Kubernetes master and worker nodes. See Create Service Accounts for Kubernetes.
- Create a load balancer to access the PKS API. See Create a Load Balancer for the PKS API.

Note: You use Ops Manager to install and configure PKS. Each version of Ops Manager supports multiple versions of PKS. To confirm that your Ops Manager version supports the version of PKS that you install, see PKS Release Notes.

Install and Configure Ops Manager

To install a compatible Ops Manager version, follow either the manual or Terraform instructions in the table below:

Version	Manual Instructions	Terraform Instructions
Ops Manager 2.3	1. Preparing to Deploy Ops Manager on GCP Manually 2. Deploying Ops Manager on GCP Manually 3. Configuring BOSH Director on GCP Manually	1. Deploying Ops Manager on GCP Using Terraform 2. Configuring BOSH Director on GCP Using Terraform
Ops Manager 2.4	1. Preparing to Deploy PCF on GCP Manually 2. Deploying BOSH and Ops Manager to GCP Manually 3. Configuring BOSH Director on GCP Manually	1. Deploying Ops Manager on GCP Using Terraform 2. Configuring BOSH Director on GCP Using Terraform

Create Service Accounts for Kubernetes

If you are installing PKS manually: After you install and configure Ops Manager, you must create service accounts for Kubernetes master and worker node VMs in your PKS deployment. To create the service accounts, follow the procedures in Creating Service Accounts in GCP for PKS.

Create a Load Balancer for the PKS API

If you are installing PKS manually: You must create an external TCP load balancer before you install PKS. This load balancer enables you to access the PKS API from outside the network and run `pks` commands from your local workstation. To create a load balancer in GCP, do the procedures in Creating a GCP Load Balancer for the PKS API.

After you install PKS, you must complete the load balancer configuration. To complete the load balancer configuration, do the procedure in Create a Network Tag for the Firewall Rule.

Resource Requirements

Installing Ops Manager and PKS requires the following virtual machines (VMs):

VM	CPU	RAM	Storage

Pivotal Container Service	2	8 GB	16 GB ^*
Pivotal Ops Manager	1	8 GB	160 GB
BOSH Director	2	8 GB	16 GB

Storage Requirements for Large Numbers of Pods

If you expect the cluster workload to run a large number of pods continuously, then increase the size of persistent disk storage allocated to the the Pivotal Container Service VM as follows:

Number of Pods	Storage (Persistent Disk) Requirement ^*
1,000 pods	20 GB
5,000 pods	100 GB
10,000 pods	200 GB
50,000 pods	1,000 GB

Kubernetes Cluster Resources

Each Kubernetes cluster provisioned through PKS deploys the VMs listed below. If you deploy more than one Kubernetes cluster, you must scale your allocated resources appropriately.

VM Name	Number	CPU Cores	RAM	Ephemeral Disk	Persistent Disk
master	1	2	4 GB	32 GB	5 GB
worker	1	2	4 GB	32 GB	50 GB

Please send any feedback you have to pks-feedback@pivotal.io.

Creating Service Accounts in GCP for PKS

Page last updated:

This topic describes the steps required to create service accounts for Pivotal Container Service (PKS) on Google Cloud Platform (GCP).

In order for Kubernetes to create load balancers and attach persistent disks to pods, you must create service accounts with sufficient permissions.

You need separate service accounts for Kubernetes cluster master and worker node VMs. Pivotal recommends configuring each service account with the least permissive privileges and unique credentials.

Create the Master Node Service Account

1. From the GCP Console, select **IAM & admin > Service accounts**.
2. Click **Create Service Account**.
3. Enter a name for the service account, and add the following roles:
    - **Compute Engine**
        - **Compute Instance Admin (v1)**
        - **Compute Network Admin**
        - **Compute Security Admin**
        - **Compute Storage Admin**
        - **Compute Viewer**
    - **Service Accounts**
        - **Service Account User**
4. Click **Create**.

Create the Worker Node Service Account

1. From the GCP Console, select **IAM & admin > Service accounts**.
2. Click **Create Service Account**.
3. Enter a name for the service account, and add the **Compute Engine > Compute Viewer** role.
4. Click **Create**.

After you create both service accounts for Kubernetes, follow the procedures in Installing PKS on GCP.

Creating a GCP Load Balancer for the PKS API

Page last updated:

This topic describes how to create a load balancer for the PKS API using Google Cloud Platform (GCP).

Overview

Before you install Pivotal Container Service (PKS), you must configure an external TCP load balancer to access the PKS API from outside the network. You can use any external TCP load balancer of your choice.

Refer to the procedures in this topic to create a load balancer using GCP. If you choose to use a different load balancer, use the configuration in this topic as a guide.

Note: This procedure uses example commands which you should modify to represent the details of your PKS installation.

To create a GCP load balancer for the PKS API, do the following:

1. Create a Load Balancer
2. Create a Firewall Rule
3. Create a DNS Entry
4. Install PKS
5. Create a Network Tag for the Firewall Rule

Create a Load Balancer

To create a load balancer using GCP, perform the following steps:

1. In a browser, navigate to the GCP console.
2. Navigate to **Network Services > Load balancing** and click **CREATE LOAD BALANCER**.
3. Under **TCP Load Balancing**, click **Start configuration**.
4. Under **Internet facing or internal only**, select **From Internet to my VMs**.
5. Under **Multiple regions or single region**, select **Single region only**.
6. Click **Continue**.
7. Name your load balancer. Pivotal recommends naming your load balancer `pks-api`.
8. Select **Backend configuration**.
    - Under **Region**, select the region where you deployed Ops Manager.
    - Under **Backends**, select **Select existing instances**. This will be automatically configured when updating the Resource Config section of the PKS tile.
    - (Optional) Under **Backup pool**, select a backup pool. If you select a backup pool, set a **Failover ratio**.
    - (Optional) Under **Health check**, select whether or not you want to create a health check.
    - Under **Session affinity**, select a session affinity configuration.
    - (Optional) Select **Advanced configurations** to configure the **Connection draining timeout**.
9. Select **Frontend configuration**.
    - (Optional) Name your frontend.
    - (Optional) Click **Add a description** and provide a description.
    - Select **Create IP address** to reserve an IP address for the PKS API endpoint.
        1. Enter a name for your reserved IP address. For example, `pks-api-ip`. GCP assigns a static IP address that appears next to the name.

    2. (Optional) Enter a description.
    3. Click **Reserve**.

  - Under **Port**, enter `9021`. Your external load balancer forwards traffic to the PKS control plane VM using the UAA endpoint on port 8443 and the PKS API endpoint on port 9021.
  - Click **Done**.
  - Click **New Frontend IP and Port**.

    1. Enter a name for the frontend IP-port mapping, such as `pks-api-uaa`.
    2. (Optional) Add a description.
    3. Under **IP** select the same static IP address that GCP assigned in the previous step.
    4. Under **Port**, enter `8443`.
    5. Click **Done**.

10. Click **Review and finalize** to review your load balancer configuration.

11. Click **Create**.

Create a Firewall Rule

To create a firewall rule that allows traffic between the load balancer and the PKS API VM, do the following:

1. From the GCP console, navigate to **VPC Network > Firewall rules** and click **CREATE FIREWALL RULE**.

2. Configure the following:

   - Name your firewall rule.
   - (Optional) Provide a description for your firewall rule.
   - Under **Network**, select the VPC network you created in the Create a GCP Network with Subnets step of *Preparing GCP*.
   - Under **Priority**, enter a priority number between `0` and `65535`.
   - Under **Direction of traffic**, select **Ingress**.
   - Under **Action on match**, select **Allow**.
   - Under **Targets**, select **Specified target tags**.
   - Under **Target tags**, enter `pks-api`.
   - Under **Source filter**, select **IP ranges**.
   - Under **Source IP ranges**, enter `0.0.0.0/0`.
   - Under **Protocols and ports**, select **Specified protocols and ports** and enter `tcp:8443,9021`.

3. Click **Create**.

Create a DNS Entry

To create a DNS entry in GCP for your PKS API domain, do the following:

1. From the GCP console, navigate to **Network Services > Cloud DNS**.

2. If you do not already have a DNS zone, click **Create zone**.

   - Provide a **Zone name** and a **DNS name**.
   - Specify whether the **DNSSEC** state of the zone is **Off**, **On**, or **Transfer**.
   - (Optional) Enter a **Description**.
   - Click **Create**.

3. Click **Add record set**.

4. Under **DNS Name**, enter a subdomain for the load balancer. For example, if your domain is `example.com`, enter `api.pks` in this field to use `api.pks.example.com` as your PKS API hostname.

5. Under **Resource Record Type**, select **A** to create a DNS address record.

6. Enter a value for **TTL** and select a **TTL Unit**.

7. Enter the static IP address that GCP assigned when you created the load balancer in Create a Load Balancer.

8. Click **Create**.

Install PKS

Follow the instructions in Installing PKS> on GCP to deploy PKS. After you finish installing PKS, continue to the Create a Network Tag for the Firewall Rule section below to complete the PKS API load balancer configuration.

Create a Network Tag for the Firewall Rule

To apply the firewall rule to the VM that hosts the PKS API, the VM must have the `pks-api` tag in GCP. Do the following:

1. From the GCP console, navigate to **Compute Engine** > **VM instances**.
2. Locate your PKS control plane VM. To locate this VM, you can search for the `pivotal-container-service` job label on the **VM instances** page.
3. Click the name of the VM to open the **VM instance details** menu.
4. Click **Edit**.
5. Verify that the **Network tags** field contains the `pks-api` tag. Add the tag if it does not appear in the field.
6. Scroll to the bottom of the screen and click **Save**.

Please send any feedback you have to pks-feedback@pivotal.io.

Installing PKS on GCP

Page last updated:

This topic describes how to install and configure Pivotal Container Service (PKS) on Google Cloud Platform (GCP).

Prerequisites

Before performing the procedures in this topic, you must have deployed and configured Ops Manager. For more information, see GCP Prerequisites and Resource Requirements.

If you use an instance of Ops Manager that you configured previously to install other runtimes, perform the following steps before you install PKS:

1. Navigate to Ops Manager.
2. Open the **Director Config** pane.
3. Select the **Enable Post Deploy Scripts** checkbox.
4. Click the **Installation Dashboard** link to return to the Installation Dashboard.
5. Click **Review Pending Changes**. Select all products you intend to deploy and review the changes. For more information, see Reviewing Pending Product Changes.
6. Click **Apply Changes**.

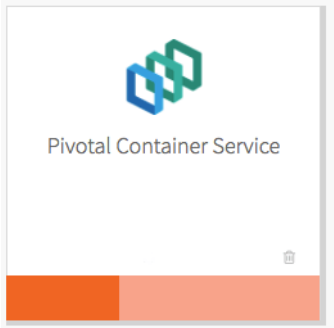
Step 1: Install PKS

To install PKS, do the following:

1. Download the product file from Pivotal Network.
2. Navigate to `https://YOUR-OPS-MANAGER-FQDN/` in a browser to log in to the Ops Manager Installation Dashboard.
3. Click **Import a Product** to upload the product file.
4. Under **Pivotal Container Service** in the left column, click the plus sign to add this product to your staging area.

Step 2: Configure PKS

Click the orange **Pivotal Container Service** tile to start the configuration process.



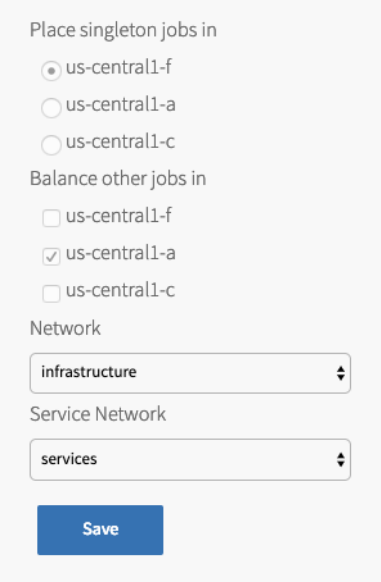
Warning: When you configure the PKS tile, do not use spaces in any field entries. This includes spaces between characters as well as leading and trailing spaces. If you use a space in any field entry, the deployment of PKS fails.

Assign AZs and Networks

Perform the following steps:

1. Click **Assign AZs and Networks**.
2. Select the availability zone (AZ) where you want to deploy the PKS API VM as a singleton job.

> **Note**: You must select an additional AZ for balancing other jobs before clicking **Save**, but this selection has no effect in the current version of PKS.

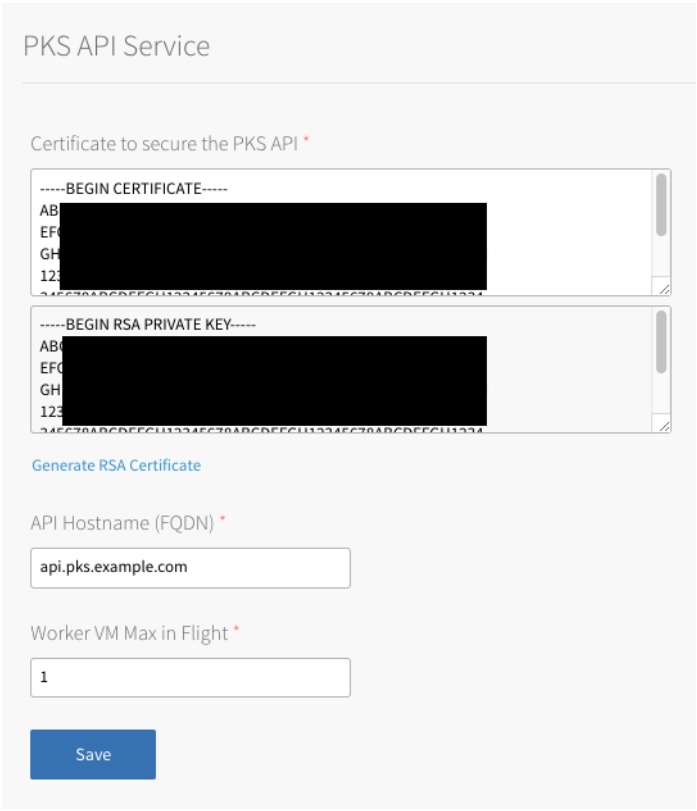


3. Under **Network**, select the infrastructure subnet that you created for the PKS API VM.
4. Under **Service Network**, select the services subnet that you created for Kubernetes cluster VMs.
5. Click **Save**.

PKS API

Perform the following steps:

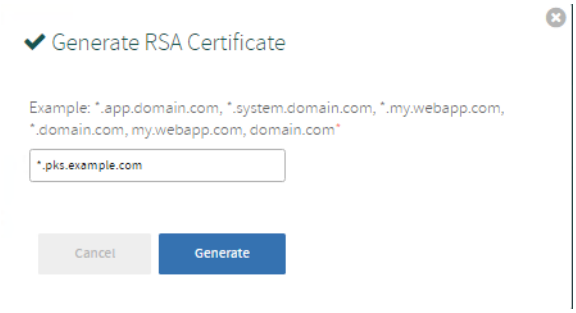
1. Click **PKS API**.
2. Under **Certificate to secure the PKS API**, provide your own certificate and private key pair.



The certificate that you supply should cover the domain that routes to the PKS API VM with TLS termination on the ingress.

If you do not have a certificate and private key pair, PKS can generate one for you. To generate a certificate, do the following:

a. Select the **Generate RSA Certificate** link.
b. Enter the domain for your API hostname. This can be a standard FQDN or a wildcard domain.
c. Click **Generate**.



Note: If you deployed a global HTTP load balancer for Ops Manager without a certificate, you can configure the load balancer to use this newly-generated certificate. To configure your Ops Manager load balancer front end certificate, see Configure Front End in *Preparing to Deploy Ops Manager on GCP Manually.*

3. Under **API Hostname (FQDN)**, enter the FQDN that you registered to point to the PKS API load balancer, such as `api.pks.example.com`. To retrieve the public IP address or FQDN of the PKS API load balancer, log in to your IaaS console.

4. Under **Worker VM Max in Flight**, enter the maximum number of non-canary worker instances to create or resize in parallel within an availability zone.

   This field sets the `max_in_flight` variable, which limits how many instances of a component can start simultaneously when a cluster is created or resized. The variable defaults to `1`, which means that only one component starts at a time.

5. Click **Save**.

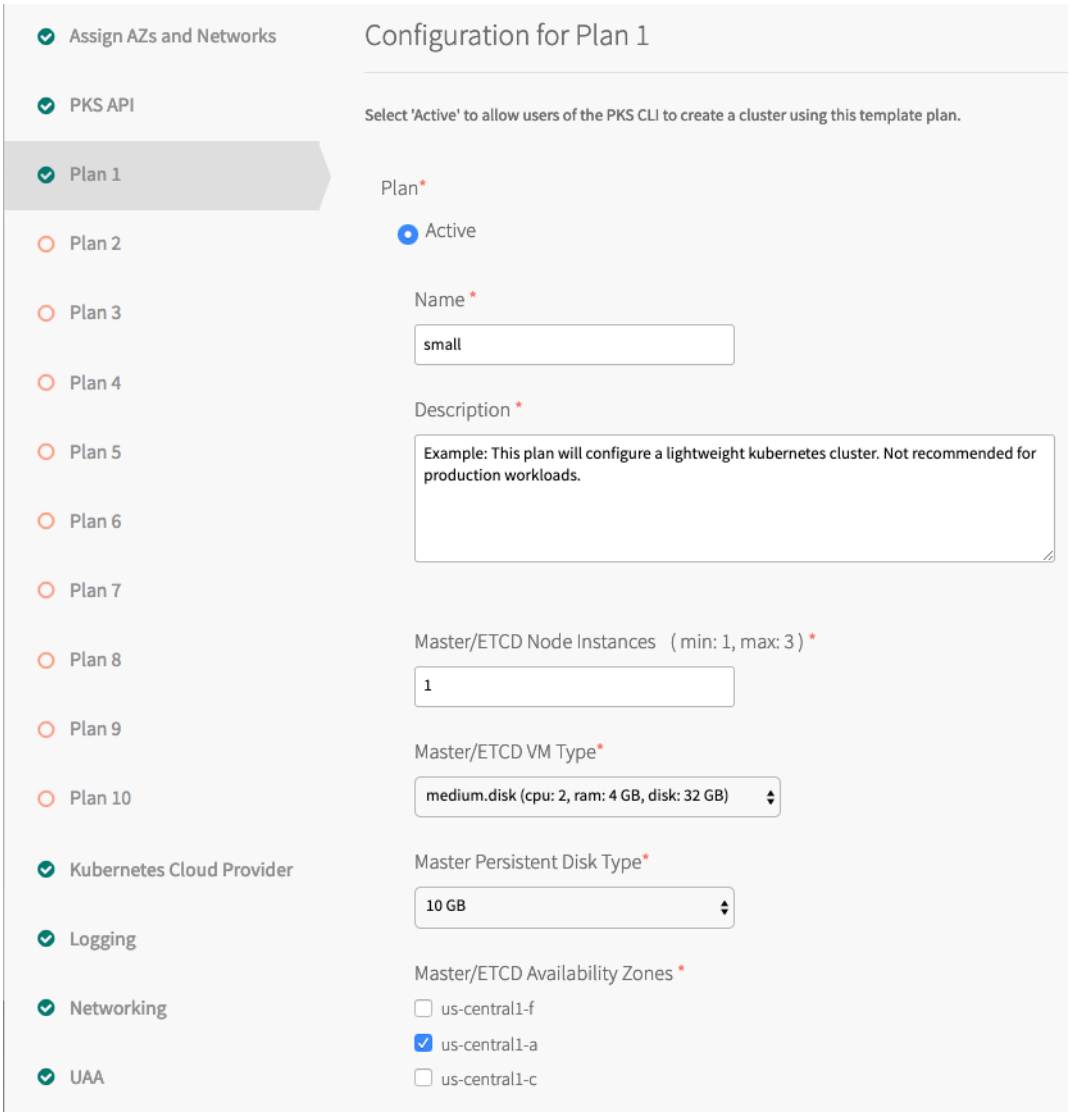
Plans

To activate a plan, perform the following steps:

1. Click the plan that you want to activate.

   Note: A plan defines a set of resource types used for deploying clusters. You can configure up to ten plans. You must configure **Plan 1**.

2. Select **Active** to activate the plan and make it available to developers deploying clusters.



3. Under **Name**, provide a unique name for the plan.

4. Under **Description**, edit the description as needed. The plan description appears in the Services Marketplace, which developers can access by using PKS CLI.

5. Under **Master/ETCD Node Instances**, select the default number of Kubernetes master/etcd nodes to provision for each cluster. You can enter either `1` or `3`.

   Note: If you deploy a cluster with multiple master/etcd node VMs, confirm that you have sufficient hardware to handle the increased load on disk write and network traffic. For more information, see Hardware recommendations in the etcd documentation.

   In addition to meeting the hardware requirements for a multi-master cluster, we recommend configuring monitoring for etcd to monitor disk latency, network latency, and other indicators for the health of the cluster. For more information, see Monitoring Master/etcd Node VMs.

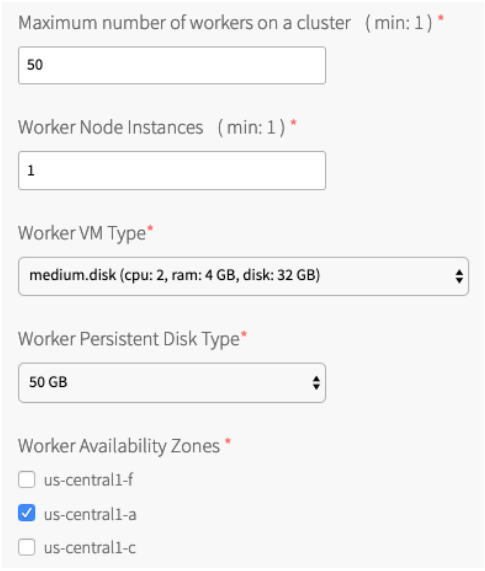
   WARNING: To change the number of master/etcd nodes for a plan, you must ensure that no existing clusters use the plan. PKS does not support changing the number of master/etcd nodes for plans with existing clusters.

6. Under **Master/ETCD VM Type**, select the type of VM to use for Kubernetes master/etcd nodes. For more information, including master node VM customization options, see the Master Node VM Size section of *VM Sizing for PKS Clusters*.

7. Under **Master Persistent Disk Type**, select the size of the persistent disk for the Kubernetes master node VM.

8. Under **Master/ETCD Availability Zones**, select one or more AZs for the Kubernetes clusters deployed by PKS. If you select more than one AZ, PKS deploys the master VM in the first AZ and the worker VMs across the remaining AZs.

9. Under **Maximum number of workers on a cluster**, set the maximum number of Kubernetes worker node VMs that PKS can deploy for each cluster. Enter any whole number in this field.



10. Under **Worker Node Instances**, select the default number of Kubernetes worker nodes to provision for each cluster.

    If the user creating a cluster with the PKS CLI does not specify a number of worker nodes, the cluster is deployed with the default number set in this field. This value cannot be greater than the maximum worker node value you set in the previous field. For more information about creating clusters, see Creating Clusters.

    For high availability, create clusters with a minimum of three worker nodes, or two per AZ if you intend to use PersistentVolumes (PVs). For example, if you deploy across three AZs, you should have six worker nodes. For more information about PVs, see PersistentVolumes in *Maintaining Workload Uptime*. Provisioning a minimum of three worker nodes, or two nodes per AZ is also recommended for stateless workloads.

    If you later reconfigure the plan to adjust the default number of worker nodes, the existing clusters that have been created from that plan are not automatically upgraded with the new default number of worker nodes.

11. Under **Worker VM Type**, select the type of VM to use for Kubernetes worker node VMs. For more information, including worker node VM customization options, see the Worker Node VM Number and Size section of *VM Sizing for PKS Clusters*.

    Note: If you install PKS in an NSX-T environment, we recommend that you select a **Worker VM Type** with a minimum disk size of 16 GB. The disk space provided by the default `medium` Worker VM Type is insufficient for PKS with NSX-T.

12. Under **Worker Persistent Disk Type**, select the size of the persistent disk for the Kubernetes worker node VMs.

13. Under **Worker Availability Zones**, select one or more AZs for the Kubernetes worker nodes. PKS deploys worker nodes equally across the AZs you select.

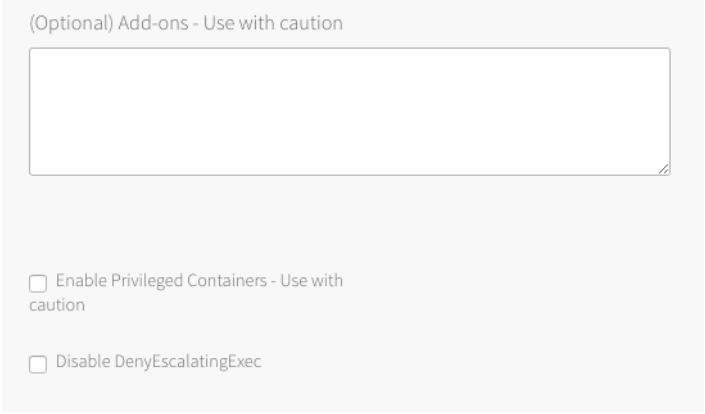
14. Under **Kubelet customization - system-reserved**, enter resource values that Kubelet can use to reserve resources for system daemons. For example, `memory=250Mi, cpu=150m`. For more information about system-reserved values, see the Kubernetes documentation.

15. Under **Kubelet customization - eviction-hard**, enter threshold limits that Kubelet can use to evict pods when they exceed the limit. Enter limits in the format `EVICTION-SIGNAL=QUANTITY`. For example, `memory.available=100Mi, nodefs.available=10%, nodefs.inodesFree=5%`. For more information about eviction thresholds, see the Kubernetes documentation.

    WARNING: Use the Kubelet customization fields with caution. If you enter values that are invalid or that exceed the limits the system supports, Kubelet might fail to start. If Kubelet fails to start, you cannot create clusters.

16. Under **Errand VM Type**, select the size of the VM that contains the errand. The smallest instance possible is sufficient, as the only errand running on this VM is the one that applies the **Default Cluster App** YAML configuration.

17. (Optional) Under **(Optional) Add-ons - Use with caution**, enter additional YAML configuration to add custom workloads to each cluster in this plan. You can specify multiple files using `---` as a separator. For more information, see Adding Custom Workloads.



18. (Optional) To allow users to create pods with privileged containers, select the **Enable Privileged Containers - Use with caution** option. For more information, see Pods in the Kubernetes documentation.

19. (Optional) To disable the admission controller, select the **Disable DenyEscalatingExec** checkbox. If you select this option, clusters in this plan can create security vulnerabilities that may impact other tiles. Use this feature with caution.

20. Click **Save**.

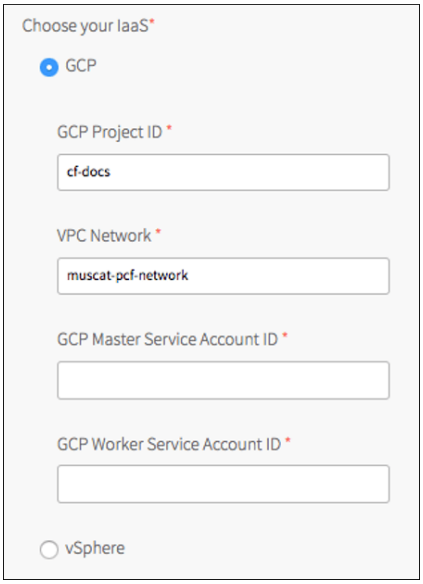
To deactivate a plan, perform the following steps:

1. Click the plan that you want to deactivate.
2. Select **Inactive**.
3. Click **Save**.

Kubernetes Cloud Provider

To configure your Kubernetes cloud provider settings, follow the procedures below:

1. Click **Kubernetes Cloud Provider**.
2. Under **Choose your IaaS**, select **GCP**.
3. Ensure the values in the following procedure match those in the **Google Config** section of the **Ops Manager** tile as follows:



a. Enter your **GCP Project ID**, which is the name of the deployment in your Ops Manager environment. To find the project ID, go to **BOSH Director for GCP** > **Google Config** > **Project ID**.
b. Enter your **VPC Network**, which is the VPC network name for your Ops Manager environment.
c. Enter your **GCP Master Service Account ID**. This is the email address associated with the master node service account.

   - **If you are installing PKS manually:** You configured the master node service account in Create the Master Node Service Account in *Creating Service Accounts in GCP for PKS*.
   - **If you are installing PKS with Terraform:** Retrieve the master node service account ID by running `terraform output` and locating the value for `pks_master_node_service_account_email`.

d. Enter your **GCP Worker Service Account ID**. This is the email address associated with the worker node service account.

   - **If you are installing PKS manually:** You configured the worker node service account in Create the Worker Node Service Account in *Creating Service Accounts in GCP for PKS*.
   - **If you are installing PKS with Terraform:** Retrieve the worker node service account ID by running `terraform output` and locating the value for `pks_worker_node_service_account_email`.

4. Click **Save**.

(Optional) Logging

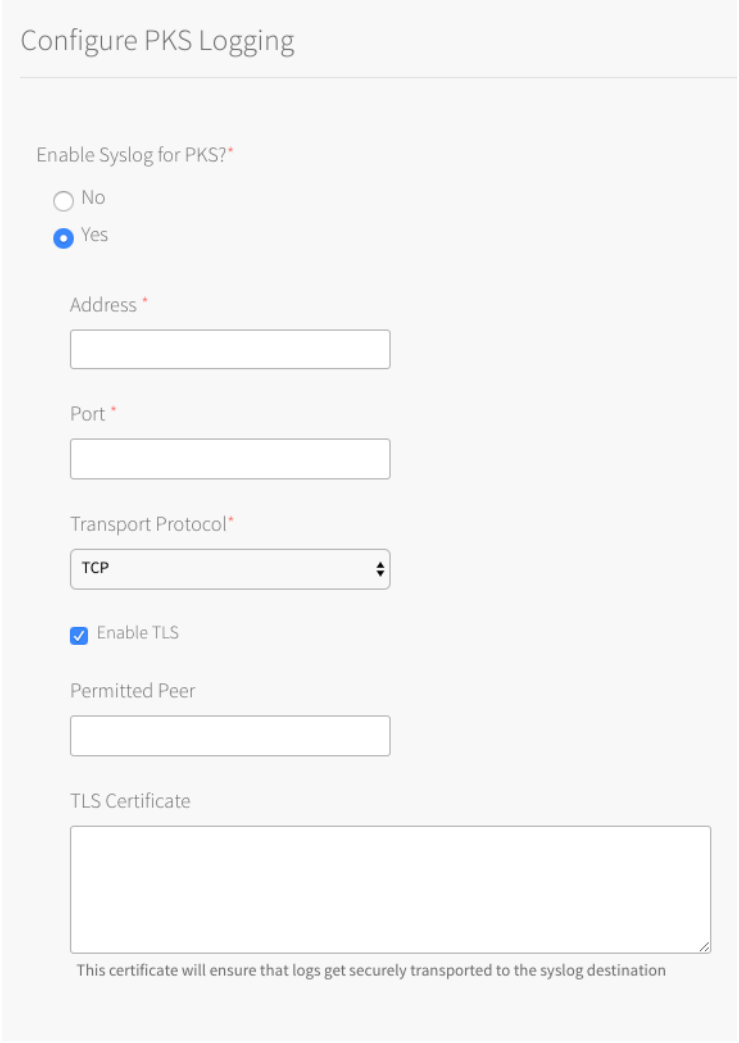
You can designate an external syslog endpoint for forwarded BOSH-deployed VM logs.

In addition, you can enable sink resources to collect PKS cluster and namespace log messages.

To configure logging in PKS, do the following:

1. Click **Logging**.

2. To enable syslog forwarding for BOSH-deployed VM logs, select **Yes**.



3. Under **Address**, enter the destination syslog endpoint.
4. Under **Port**, enter the destination syslog port.
5. Select a transport protocol for log forwarding.
6. (Optional) Pivotal strongly recommends that you enable TLS encryption when forwarding logs as they may contain sensitive information. For example, these logs may contain cloud provider credentials. To enable TLS, perform the following steps:
    a. Under **Permitter Peer**, provide the accepted fingerprint (SHA1) or name of remote peer. For example, `*.YOUR-LOGGING-SYSTEM.com`.
    b. Under **TLS Certificate**, provide a TLS certificate for the destination syslog endpoint.

    > **Note:** You do not need to provide a new certificate if the TLS certificate for the destination syslog endpoint is signed by a Certificate Authority (CA) in your BOSH certificate store.

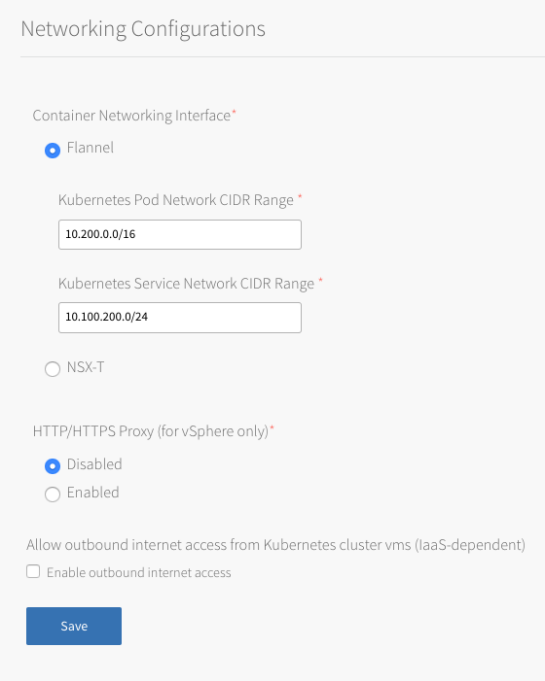
7. To enable clusters to drain Kubernetes API events and pod logs to sinks using `syslog://`, select **Enable Sink Resources**. For more information about using sink resources, see Creating Sink Resources.

8. Click **Save**.

Networking

To configure networking, do the following:

1. Click **Networking**.

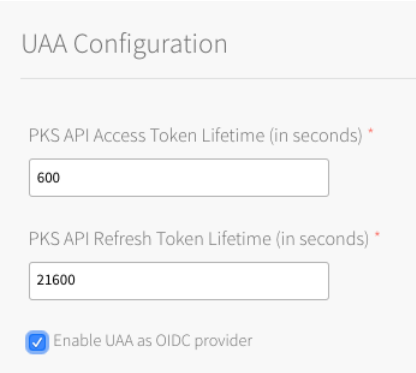


2. Under **Container Networking Interface**, select **Flannel**.
3. (Optional) Enter values for **Kubernetes Pod Network CIDR Range** and **Kubernetes Service Network CIDR Range**.
    - Ensure that the CIDR ranges do not overlap and have sufficient space for your deployed services.
    - Ensure that the CIDR range for the **Kubernetes Pod Network CIDR Range** is large enough to accommodate the expected maximum number of pods.
4. (Optional) If you do not use a NAT instance, select **Allow outbound internet access from Kubernetes cluster vms (IaaS-dependent)**. Enabling this functionality assigns external IP addresses to VMs in clusters.
5. Click **Save**.

UAA

To configure the UAA server, do the following:

1. Click **UAA**.
2. Under **PKS API Access Token Lifetime**, enter a time in seconds for the PKS API access token lifetime.



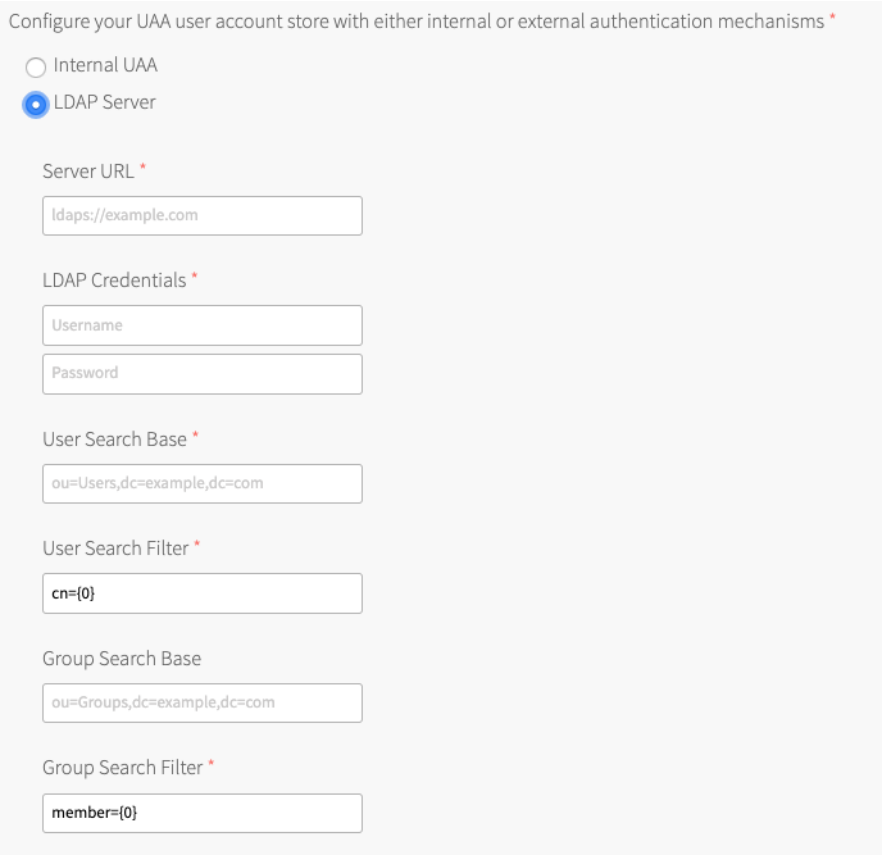
3. Under **PKS API Refresh Token Lifetime**, enter a time in seconds for the PKS API refresh token lifetime.
4. Select one of the following options:
    - To use an internal user account store for UAA, select **Internal UAA**. Click **Save** and continue to (Optional) Monitoring.
    - To use an external user account store for UAA, select **LDAP Server** and continue to Configure LDAP as an Identity Provider.

> **Note**: Selecting **LDAP Server** allows admin users to give cluster access to groups of users. For more information about performing this procedure, see Grant Cluster Access to a Group in *Managing Users in PKS with UAA*.

Configure LDAP as an Identity Provider

To integrate UAA with one or more LDAP servers, configure PKS with your LDAP endpoint information as follows:

1. Under **UAA**, select **LDAP Server**.



2. For **Server URL**, enter the URLs that point to your LDAP server. If you have multiple LDAP servers, separate their URLs with spaces. Each URL must include one of the following protocols:
    - `ldap://` : Use this protocol if your LDAP server uses an unencrypted connection.

- `ldaps://`: Use this protocol if your LDAP server uses SSL for an encrypted connection. To support an encrypted connection, the LDAP server must hold a trusted certificate or you must import a trusted certificate to the JVM truststore.

3. For **LDAP Credentials**, enter the LDAP Distinguished Name (DN) and password for binding to the LDAP server. For example, `cn=administrator,ou=Users,dc=example,dc=com`. If the bind user belongs to a different search base, you must use the full DN.

   Note: We recommend that you provide LDAP credentials that grant read-only permissions on the LDAP search base and the LDAP group search base.

4. For **User Search Base**, enter the location in the LDAP directory tree where LDAP user search begins. The LDAP search base typically matches your domain name.

   For example, a domain named `cloud.example.com` may use `ou=Users,dc=example,dc=com` as its LDAP user search base.

5. For **User Search Filter**, enter a string to use for LDAP user search criteria. The search criteria allows LDAP to perform more effective and efficient searches. For example, the standard LDAP search filter `cn=Smith` returns all objects with a common name equal to `Smith`.

   In the LDAP search filter string that you use to configure PKS, use `{0}` instead of the username. For example, use `cn={0}` to return all LDAP objects with the same common name as the username.

   In addition to `cn`, other common attributes are `mail`, `uid` and, in the case of Active Directory, `sAMAccountName`.

   Note: For information about testing and troubleshooting your LDAP search filters, see Configuring LDAP Integration with Pivotal Cloud Foundry.

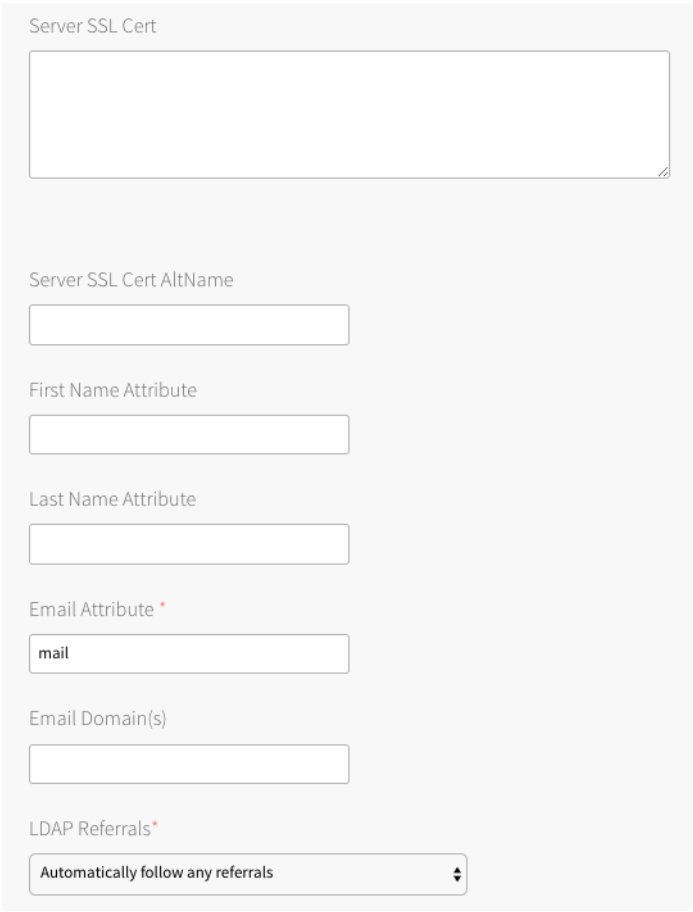
6. For **Group Search Base**, enter the location in the LDAP directory tree where the LDAP group search begins.

   For example, a domain named `cloud.example.com` may use `ou=Groups,dc=example,dc=com` as its LDAP group search base.

   Follow the instructions in the Grant PKS Access to an External LDAP Group section of *Managing Users in PKS with UAA* to map the groups under this search base to roles in PKS.

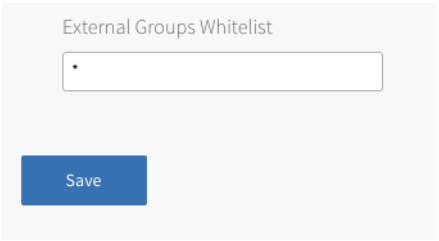
7. For **Group Search Filter**, enter a string that defines LDAP group search criteria. The standard value is `member={0}`.

8. For **Server SSL Cert**, paste in the root certificate from your CA certificate or your self-signed certificate.



9. For **Server SSL Cert AltName**, do one of the following:
    - If you are using `ldaps://` with a self-signed certificate, enter a Subject Alternative Name (SAN) for your certificate.
    - If you are not using `ldaps://` with a self-signed certificate, leave this field blank.
10. For **First Name Attribute**, enter the attribute name in your LDAP directory that contains user first names. For example, `cn`.
11. For **Last Name Attribute**, enter the attribute name in your LDAP directory that contains user last names. For example, `sn`.
12. For **Email Attribute**, enter the attribute name in your LDAP directory that contains user email addresses. For example, `mail`.
13. For **Email Domain(s)**, enter a comma-separated list of the email domains for external users who can receive invitations to Apps Manager.
14. For **LDAP Referrals**, choose how UAA handles LDAP server referrals to other user stores. UAA can follow the external referrals, ignore them without returning errors, or generate an error for each external referral and abort the authentication.
15. For **External Groups Whitelist**, enter a comma-separated list of group patterns which need to be populated in the user's `id_token`. For further information on accepted patterns see the description of the `config.externalGroupsWhitelist` in the OAuth/OIDC Identity Provider Documentation.

    Note: When sent as a Bearer token in the Authentication header, wide pattern queries for users who are members of multiple groups, can cause the size of the `id_token` to extend beyond what is supported by web servers.

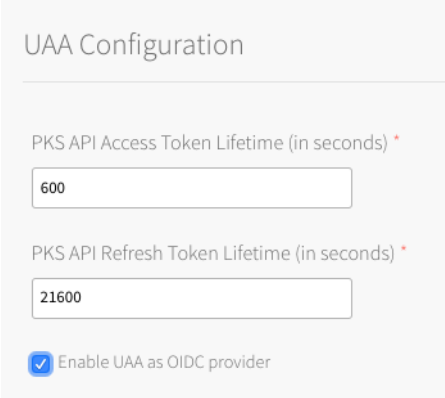


16. Click **Save**.

(Optional) Configure OpenID Connect

You can use OpenID Connect (OIDC) to instruct Kubernetes to verify end-user identities based on authentication performed by an authorization server, such as UAA.

To configure PKS to use OIDC, select **Enable UAA as OIDC provider**. With OIDC enabled, Admin Users can grant cluster-wide access to Kubernetes end users.



For more information about configuring OIDC, see the table below:

Option	Description
OIDC disabled	If you do not enable OIDC, Kubernetes authenticates users against its internal user management system.
OIDC enabled	If you enable OIDC, Kubernetes uses the authentication mechanism that you selected in UAA as follows: • If you selected **Internal UAA**, Kubernetes authenticates users against the internal UAA authentication mechanism. • If you selected **LDAP Server**, Kubernetes authenticates users against the LDAP server.

For additional information about getting credentials with OIDC configured, see Retrieve Cluster Credentials in *Retrieving Cluster Credentials and Configuration*.

Note: When you enable OIDC, existing PKS-provisioned Kubernetes clusters are upgraded to use OIDC. This invalidates your kubeconfig files. You must regenerate the files for all clusters.

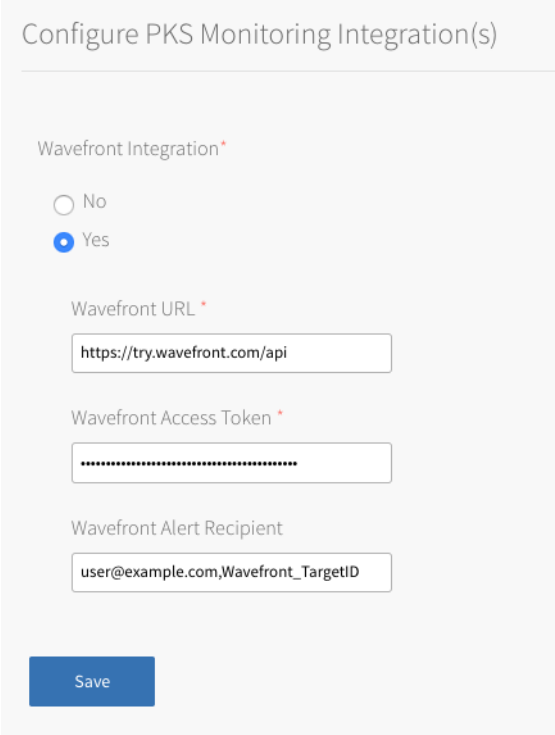
(Optional) Monitoring

You can monitor Kubernetes clusters and pods metrics externally using the integration with Wavefront by VMware.

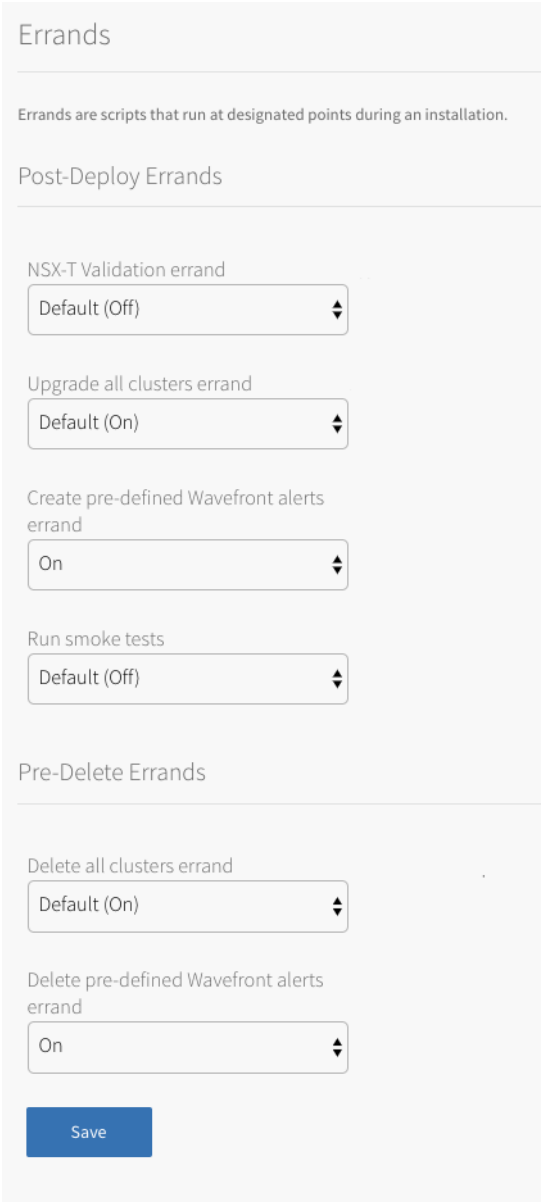
Note: Before you configure Wavefront integration, you must have an active Wavefront account and access to a Wavefront instance. You provide your Wavefront access token during configuration and enabling errands. For additional information, see Pivotal Container Service Integration Details in the Wavefront documentation.

By default, monitoring is disabled. To enable and configure Wavefront monitoring, do the following:

1. Select **Monitoring**.



2. On the **Monitoring** pane, under **Wavefront Integration**, select **Yes**.
3. Under **Wavefront URL**, enter the URL of your Wavefront subscription. For example, `https://try.wavefront.com/api`.
4. Under **Wavefront Access Token**, enter the API token for your Wavefront subscription.
5. To configure Wavefront to send alerts by email, enter email addresses or Wavefront Target IDs separated by commas under **Wavefront Alert Recipient**. For example, `user@example.com,Wavefront_TargetID`. To create alerts, you must enable errands.
6. Select **Errands**.
7. On the **Errands** pane, enable **Create pre-defined Wavefront alerts errand**.



8. Enable **Delete pre-defined Wavefront alerts errand**.
9. Click **Save**. Your settings apply to any clusters created after you have saved these configuration settings and clicked **Apply Changes**.

> **Note**: The PKS tile does not validate your Wavefront configuration settings. To verify your setup, look for cluster and pod metrics in Wavefront.

Usage Data

VMware's Customer Experience Improvement Program (CEIP) and the Pivotal Telemetry Program (Telemetry) provides VMware and Pivotal with information that enables the companies to improve their products and services, fix problems, and advise you on how best to deploy and use our products. As part of the CEIP and Telemetry, VMware and Pivotal collect technical information about your organization's use of the Pivotal Container Service (PKS) on a regular basis. Since PKS is jointly developed and sold by VMware and Pivotal, we will share this information with one another. Information collected under CEIP or Telemetry does not personally identify any individual.

Regardless of your selection in the **Usage Data** pane, a small amount of data is sent from Cloud Foundry Container Runtime (CFCR) to the PKS tile. However, that data is not shared externally.

To configure the **Usage Data** pane, perform the following steps:

1. Select the **Usage Data** side-tab.
2. Read the Usage Data description.

3. Make your selection.

    a. To join the program, select **Yes, I want to join the CEIP and Telemetry Program for PKS**.
    b. To decline joining the program, select **No, I do not want to join the CEIP and Telemetry Program for PKS**.

4. Click **Save**.

Note: If you join the CEIP and Telemetry Program for PKS, open your firewall to allow outgoing access to `https://vcsa.vmware.com/ph-prd` on port `443`.

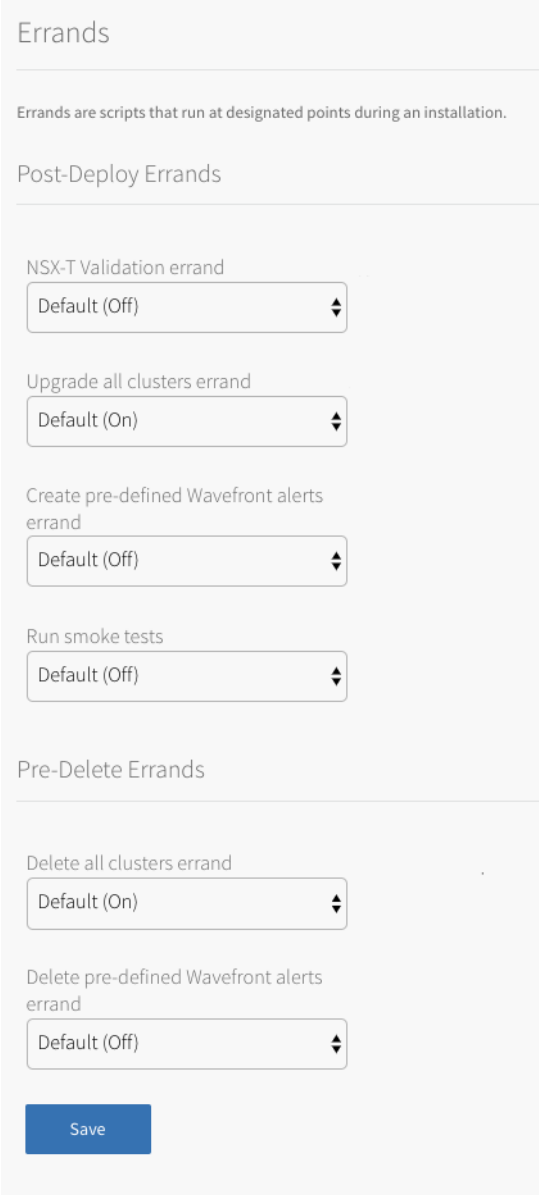
Errands

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

To configure when post-deploy and pre-delete errands for PKS are run, make a selection in the dropdown next to the errand.

We recommend that you set the **Run smoke tests** errand to **On**. The errand uses the PKS Command Line Interface (PKS CLI) to create a Kubernetes cluster and then delete it. If the creation or deletion fails, the errand fails and the installation of the PKS tile is aborted.

For the other errands, we recommend that you leave the default settings.



For more information about errands and their configuration state, see Managing Errands in Ops Manager.

WARNING: Because PKS uses floating stemcells, updating the PKS tile with a new stemcell triggers the rolling of every VM in each cluster. Also, updating other product tiles in your deployment with a new stemcell causes the PKS tile to roll VMs. This rolling is enabled by the **Upgrade all clusters errand**. We recommend that you keep this errand turned on because automatic rolling of VMs ensures that all deployed cluster VMs are patched. However, automatic rolling can cause downtime in your deployment.

If you are upgrading PKS, you must enable the **Upgrade All Clusters** errand.

Resource Config

To modify the resource usage of PKS and specify your PKS API load balancer, follow the steps below:

1. Select **Resource Config**.
2. In the **Load Balancers** column, enter the name of your PKS API load balancer, prefixed with `tcp:`. For example:

   ```
   tcp:PKS-API-LB
   ```

   Where `PKS-API-LB` is the name of your PKS API load balancer.

   You can find the name of your PKS API load balancer by doing one of the following:

   - **If you are installing PKS manually:** The name of your PKS API load balancer is the name you configured in the Create a Load Balancer section of *Creating a GCP Load Balancer for the PKS API*.
   - **If you are installing PKS using Terraform:** The name of your PKS API load balancer is the value of `pks_lb_backend_name` from `terraform output`.

   Note: After you click **Apply Changes** for the first time, BOSH assigns the PKS VM an IP address. BOSH uses the name you provide in the **Load Balancers** column to locate your load balancer, and then connect the load balancer to the PKS VM using its new IP address.

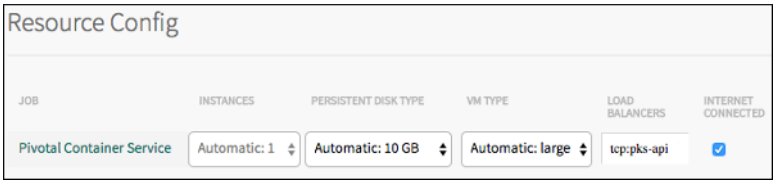
VMs used by **Pivotal Container Service** jobs must meet the following minimum requirements:

CPU	Memory	Disk
2	8 GB	29 GB

Note: If you experience timeouts or slowness when interacting with the PKS API, select a **VM Type** with greater CPU and memory resources.

To confirm you are deploying **Pivotal Container Service** job VMs meeting the minimum requirements, perform the following steps:

1. Select a **VM Type** with CPU, memory and disk resources either matching or exceeding the minimum **Pivotal Container Service** job VM requirements.



2. Select **Save**.

Step 3: Apply Changes

1. Return to the Ops Manager Installation Dashboard.
2. Click **Review Pending Changes**. Select the product that you intend to deploy and review the changes. For more information, see Reviewing Pending Product Changes.
3. Click **Apply Changes**.

Step 4: Retrieve the PKS API Endpoint

You must share the PKS API endpoint to allow your organization to use the API to create, update, and delete clusters. For more information, see Creating Clusters.

To retrieve the PKS API endpoint, do the following:

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the Pivotal Container Service tile.
3. Click the **Status** tab and locate the **Pivotal Container Service** job. The IP address of the Pivotal Container Service job is the PKS API endpoint.

Step 5: Configure External Load Balancer

If you are installing PKS manually, follow the procedure in the Create a Network Tag for the Firewall Rule section of *Creating a GCP Load Balancer for the PKS API*.

Step 6: Install the PKS and Kubernetes CLIs

The PKS and Kubernetes CLIs help you interact with your PKS-provisioned Kubernetes clusters and Kubernetes workloads. To install the CLIs, follow the instructions below:

- Installing the PKS CLI
- Installing the Kubernetes CLI

Step 7: Configure PKS API Access

Follow the procedures in Configuring PKS API Access.

Step 8: Configure Authentication for PKS

Configure authentication for PKS using User Account and Authentication (UAA). For information, see Managing Users in PKS with UAA.

Next Steps

After installing PKS on GCP, you may want to do one or more of the following:

- Create a load balancer for your PKS clusters. For more information, see Creating and Configuring a GCP Load Balancer for PKS Clusters.
- Create your first PKS cluster. For more information, see Creating Clusters.

Please send any feedback you have to pks-feedback@pivotal.io.

Amazon Web Services (AWS)

This topic outlines the steps for installing Pivotal Container Service (PKS) on Amazon Web Services (AWS). See the following sections:

Note: The topics below provide the Terraform procedures for deploying Ops Manager on AWS, not the manual procedures. The Terraform procedures are the currently supported path for deploying Ops Manager on AWS for use with PKS.

- AWS Prerequisites and Resource Requirements
- Deploying Ops Manager on AWS:
  - Deploying Ops Manager v2.3 on AWS Using Terraform or
  - Deploying Ops Manager v2.4 on AWS Using Terraform
- Configuring Ops Manager on AWS:
  - Configuring BOSH Director v2.3 on AWS Using Terraform or
  - Configuring BOSH Director v2.4 on AWS Using Terraform
- Installing PKS on AWS
- Installing the PKS CLI
- Installing the Kubernetes CLI

Please send any feedback you have to pks-feedback@pivotal.io.

AWS Prerequisites and Resource Requirements

Page last updated:

This topic describes the prerequisites and resource requirements for installing Pivotal Container Service (PKS) on Amazon Web Services (AWS).

Prerequisites

Before you install PKS, you must install one of the following:

- Ops Manager v2.3.1 or later
- Ops Manager v2.4.x

Note: You use Ops Manager to install and configure PKS. Each version of Ops Manager supports multiple versions of PKS. To confirm that your Ops Manager version supports the version of PKS that you install, see PKS Release Notes.

To install an Ops Manager version that is compatible with the PKS version you intend to use, follow the instructions in the corresponding version of the Ops Manager documentation.

Note: The topics below provide the Terraform procedures for deploying Ops Manager on AWS, not the manual procedures. The Terraform procedures are the currently supported path for deploying Ops Manager on AWS for use with PKS.

Version	
Ops Manager v2.3	• Deploying Ops Manager on AWS Using Terraform • Configuring BOSH Director on AWS Using Terraform
Ops Manager v2.4	• Deploying Ops Manager on AWS Using Terraform • Configuring BOSH Director on AWS Using Terraform

Resource Requirements

Installing Ops Manager and PKS requires the following virtual machines (VMs):

VM Name	VM Type	Default VM Count
Pivotal Container Service	m4.large ^*	1
BOSH Director	m4.large	1

Storage Requirements for Large Numbers of Pods

If you expect the cluster workload to run a large number of pods continuously, then increase the size of persistent disk storage allocated to the the Pivotal Container Service VM as follows:

Number of Pods	Storage (Persistent Disk) Requirement ^*
1,000 pods	20 GB
5,000 pods	100 GB
10,000 pods	200 GB
50,000 pods	1,000 GB

Kubernetes Cluster Resources

Each Kubernetes cluster provisioned through PKS deploys the VMs listed below. If you deploy more than one Kubernetes cluster, you must scale your

allocated resources appropriately.

VM Name	Number	CPU Cores	RAM	Ephemeral Disk	Persistent Disk
master	1	2	4 GB	32 GB	5 GB
worker	1	2	4 GB	32 GB	50 GB

Please send any feedback you have to pks-feedback@pivotal.io.

Installing PKS on AWS

Page last updated:

This topic describes how to install and configure Pivotal Container Service (PKS) on Amazon Web Services (AWS).

Prerequisites

Before performing the procedures in this topic, you must have deployed and configured Ops Manager. For more information, see AWS Prerequisites and Resource Requirements.

This topic assumes that you used Terraform to prepare the AWS environment for this Pivotal Container Service (PKS) deployment. You retrieve specific values required by this deployment by running `terraform output`.

For more information, see Deploying Ops Manager on AWS Using Terraform.

If you use an instance of Ops Manager that you configured previously to install other runtimes, perform the following steps before you install PKS:

1. Navigate to Ops Manager.
2. Open the **Director Config** pane.
3. Select the **Enable Post Deploy Scripts** checkbox.
4. Click the **Installation Dashboard** link to return to the Installation Dashboard.
5. Click **Review Pending Changes**. Select all products you intend to deploy and review the changes. For more information, see Reviewing Pending Product Changes.
6. Click **Apply Changes**.

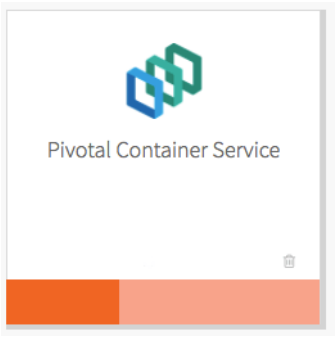
Step 1: Install PKS

To install PKS, do the following:

1. Download the product file from Pivotal Network.
2. Navigate to `https://YOUR-OPS-MANAGER-FQDN/` in a browser to log in to the Ops Manager Installation Dashboard.
3. Click **Import a Product** to upload the product file.
4. Under **Pivotal Container Service** in the left column, click the plus sign to add this product to your staging area.

Step 2: Configure PKS

Click the orange **Pivotal Container Service** tile to start the configuration process.



WARNING: When you configure the PKS tile, do not use spaces in any field entries. This includes spaces between characters as well as leading and

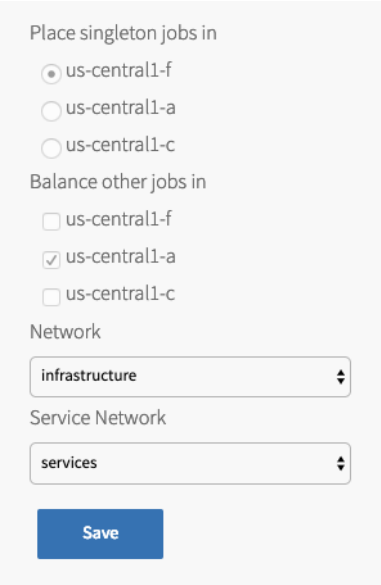
trailing spaces. If you use a space in any field entry, the deployment of PKS fails.

Assign AZs and Networks

Perform the following steps:

1. Click **Assign AZs and Networks**.
2. Select the availability zone (AZ) where you want to deploy the PKS API VM as a singleton job.

> **Note**: You must select an additional AZ for balancing other jobs before clicking **Save**, but this selection has no effect in the current version of PKS.

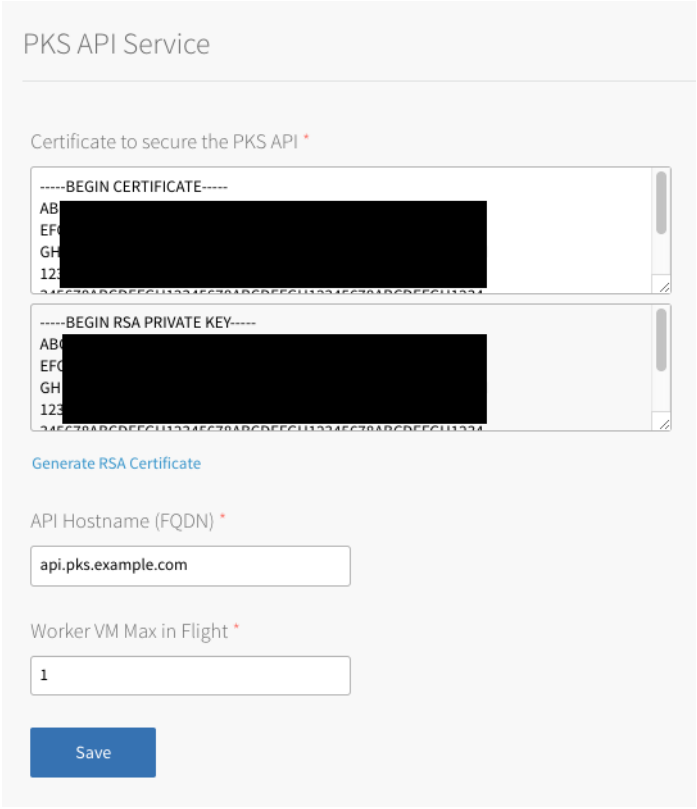


3. Under **Network**, select the infrastructure subnet that you created for the PKS API VM.
4. Under **Service Network**, select the services subnet that you created for Kubernetes cluster VMs.
5. Click **Save**.

PKS API

Perform the following steps:

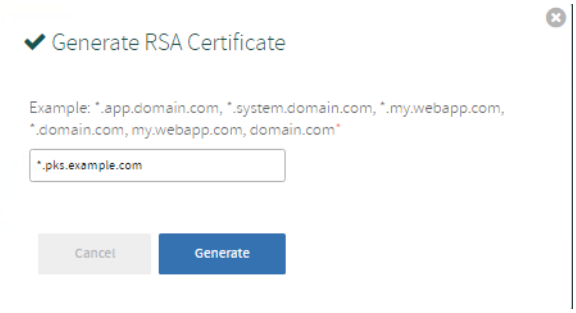
1. Click **PKS API**.
2. Under **Certificate to secure the PKS API**, provide your own certificate and private key pair.



The certificate that you supply should cover the domain that routes to the PKS API VM with TLS termination on the ingress.

If you do not have a certificate and private key pair, PKS can generate one for you. To generate a certificate, do the following:

a. Select the **Generate RSA Certificate** link.
b. Enter the domain for your API hostname. This can be a standard FQDN or a wildcard domain.
c. Click **Generate**.



3. Under **API Hostname (FQDN)**, enter the FQDN that you registered to point to the PKS API load balancer, such as `api.pks.example.com`. To retrieve the public IP address or FQDN of the PKS API load balancer, see the `terraform.tfstate` file.

4. Under **Worker VM Max in Flight**, enter the maximum number of non-canary worker instances to create or resize in parallel within an availability zone.

   This field sets the `max_in_flight` variable, which limits how many instances of a component can start simultaneously when a cluster is created or resized. The variable defaults to `1`, which means that only one component starts at a time.

5. Click **Save**.

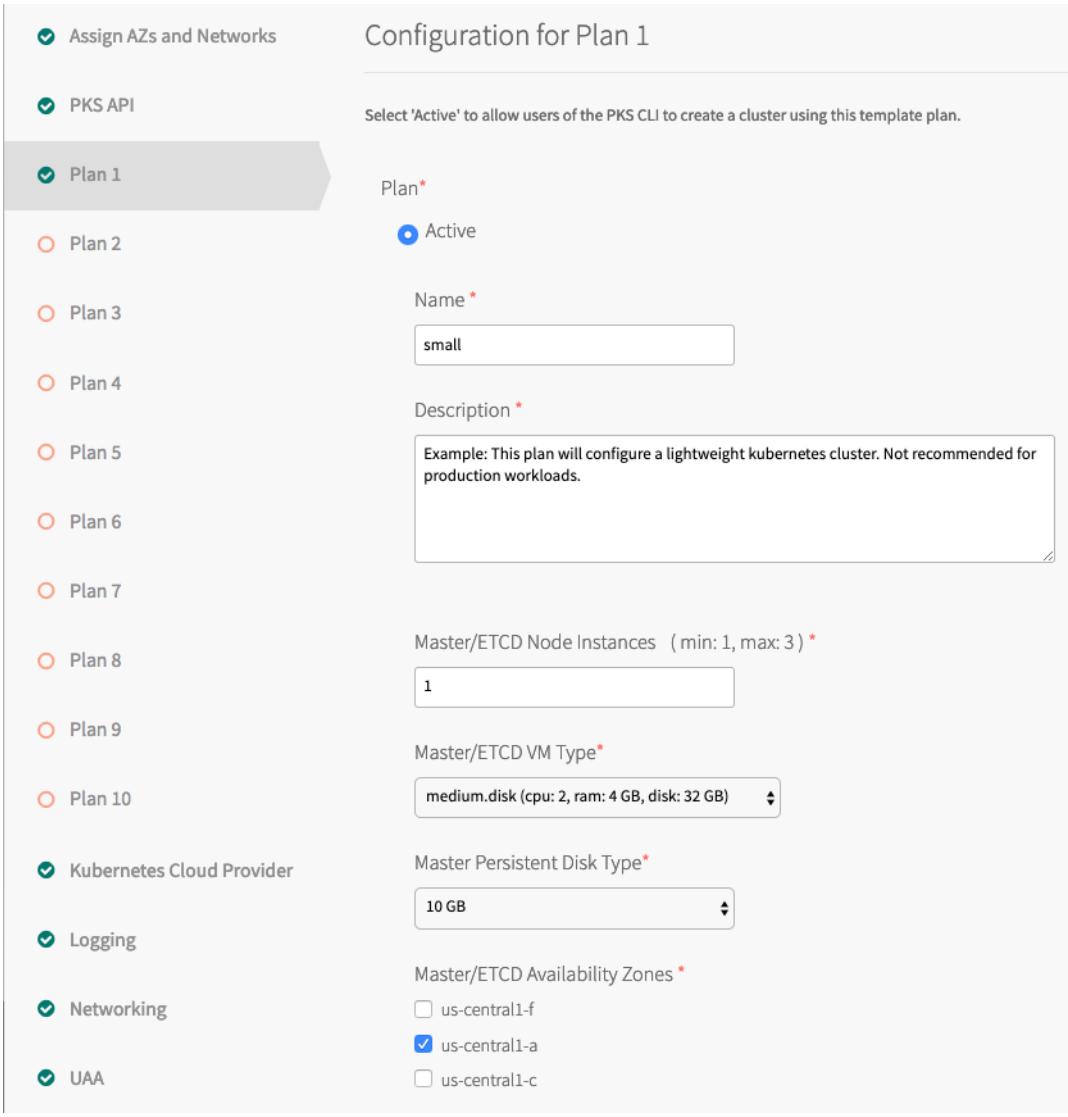
Plans

To activate a plan, perform the following steps:

1. Click the plan that you want to activate.

> **Note**: A plan defines a set of resource types used for deploying clusters. You can configure up to ten plans. You must configure **Plan 1**.

2. Select **Active** to activate the plan and make it available to developers deploying clusters.



3. Under **Name**, provide a unique name for the plan.

4. Under **Description**, edit the description as needed. The plan description appears in the Services Marketplace, which developers can access by using PKS CLI.

5. Under **Master/ETCD Node Instances**, select the default number of Kubernetes master/etcd nodes to provision for each cluster. You can enter either `1` or `3`.

> **Note**: If you deploy a cluster with multiple master/etcd node VMs, confirm that you have sufficient hardware to handle the increased load on disk write and network traffic. For more information, see Hardware recommendations in the etcd documentation.
>
> In addition to meeting the hardware requirements for a multi-master cluster, we recommend configuring monitoring for etcd to monitor disk latency, network latency, and other indicators for the health of the cluster. For more information, see Monitoring Master/etcd Node VMs.

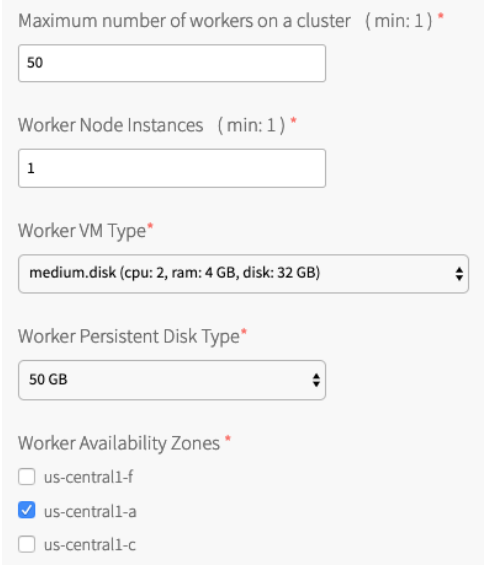
> **WARNING**: To change the number of master/etcd nodes for a plan, you must ensure that no existing clusters use the plan. PKS does not support changing the number of master/etcd nodes for plans with existing clusters.

6. Under **Master/ETCD VM Type**, select the type of VM to use for Kubernetes master/etcd nodes. For more information, including master node VM customization options, see the Master Node VM Size section of *VM Sizing for PKS Clusters*.

7. Under **Master Persistent Disk Type**, select the size of the persistent disk for the Kubernetes master node VM.

8. Under **Master/ETCD Availability Zones**, select one or more AZs for the Kubernetes clusters deployed by PKS. If you select more than one AZ, PKS deploys the master VM in the first AZ and the worker VMs across the remaining AZs.

9. Under **Maximum number of workers on a cluster**, set the maximum number of Kubernetes worker node VMs that PKS can deploy for each cluster. Enter any whole number in this field.



10. Under **Worker Node Instances**, select the default number of Kubernetes worker nodes to provision for each cluster.

    If the user creating a cluster with the PKS CLI does not specify a number of worker nodes, the cluster is deployed with the default number set in this field. This value cannot be greater than the maximum worker node value you set in the previous field. For more information about creating clusters, see Creating Clusters.

    For high availability, create clusters with a minimum of three worker nodes, or two per AZ if you intend to use PersistentVolumes (PVs). For example, if you deploy across three AZs, you should have six worker nodes. For more information about PVs, see PersistentVolumes in *Maintaining Workload Uptime*. Provisioning a minimum of three worker nodes, or two nodes per AZ is also recommended for stateless workloads.

    If you later reconfigure the plan to adjust the default number of worker nodes, the existing clusters that have been created from that plan are not automatically upgraded with the new default number of worker nodes.

11. Under **Worker VM Type**, select the type of VM to use for Kubernetes worker node VMs. For more information, including worker node VM customization options, see the Worker Node VM Number and Size section of *VM Sizing for PKS Clusters*.

    Note: If you install PKS in an NSX-T environment, we recommend that you select a **Worker VM Type** with a minimum disk size of 16 GB. The disk space provided by the default `medium` Worker VM Type is insufficient for PKS with NSX-T.

12. Under **Worker Persistent Disk Type**, select the size of the persistent disk for the Kubernetes worker node VMs.

13. Under **Worker Availability Zones**, select one or more AZs for the Kubernetes worker nodes. PKS deploys worker nodes equally across the AZs you select.

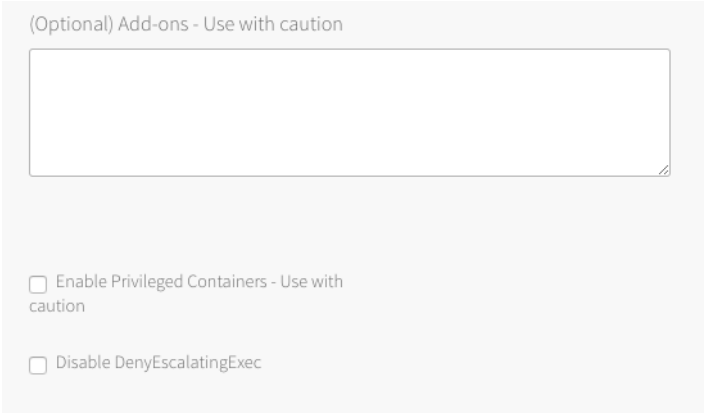
14. Under **Kubelet customization - system-reserved**, enter resource values that Kubelet can use to reserve resources for system daemons. For example, `memory=250Mi, cpu=150m`. For more information about system-reserved values, see the Kubernetes documentation.

15. Under **Kubelet customization - eviction-hard**, enter threshold limits that Kubelet can use to evict pods when they exceed the limit. Enter limits in the format `EVICTION-SIGNAL=QUANTITY`. For example, `memory.available=100Mi, nodefs.available=10%, nodefs.inodesFree=5%`. For more information about eviction thresholds, see the Kubernetes documentation.

    WARNING: Use the Kubelet customization fields with caution. If you enter values that are invalid or that exceed the limits the system supports, Kubelet might fail to start. If Kubelet fails to start, you cannot create clusters.

16. Under **Errand VM Type**, select the size of the VM that contains the errand. The smallest instance possible is sufficient, as the only errand running on this VM is the one that applies the **Default Cluster App** YAML configuration.

17. (Optional) Under **(Optional) Add-ons - Use with caution**, enter additional YAML configuration to add custom workloads to each cluster in this plan. You can specify multiple files using `---` as a separator. For more information, see Adding Custom Workloads.



18. (Optional) To allow users to create pods with privileged containers, select the **Enable Privileged Containers - Use with caution** option. For more information, see Pods in the Kubernetes documentation.
19. (Optional) To disable the admission controller, select the **Disable DenyEscalatingExec** checkbox. If you select this option, clusters in this plan can create security vulnerabilities that may impact other tiles. Use this feature with caution.
20. Click **Save**.

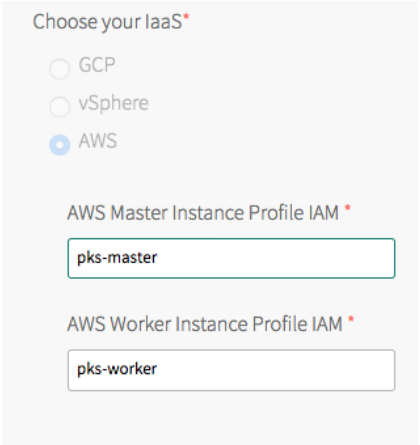
To deactivate a plan, perform the following steps:

1. Click the plan that you want to deactivate.
2. Select **Inactive**.
3. Click **Save**.

Kubernetes Cloud Provider

To configure your Kubernetes cloud provider settings, follow the procedures below:

1. Click **Kubernetes Cloud Provider**.
2. Under **Choose your IaaS**, select **AWS**.



3. Enter your **AWS Master Instance Profile IAM**. This is the instance profile name associated with the master node. To retrieve the instance profile name, run `terraform output` and locate the value for the field `pks_master_iam_instance_profile_name`.
4. Enter your **AWS Worker Instance Profile IAM**. This is the instance profile name associated with the worker node. To retrieve the instance profile name, run `terraform output` and locate the value for the field `pks_worker_iam_instance_profile_name`.
5. Click **Save**.

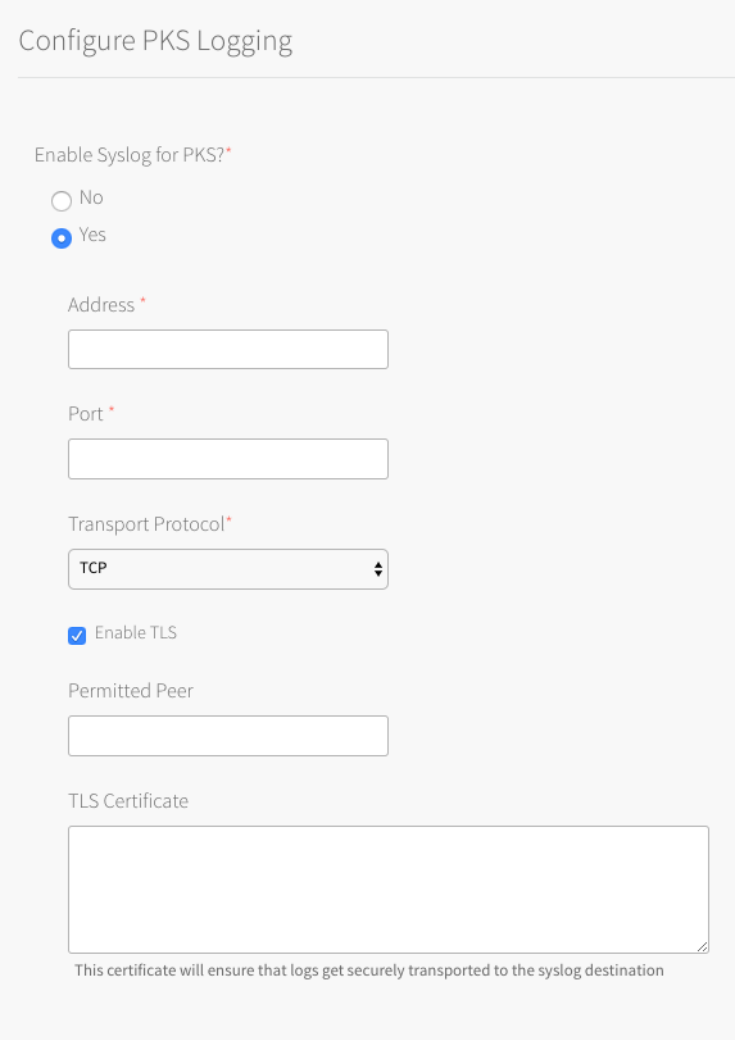
(Optional) Logging

You can designate an external syslog endpoint for forwarded BOSH-deployed VM logs.

In addition, you can enable sink resources to collect PKS cluster and namespace log messages.

To configure logging in PKS, do the following:

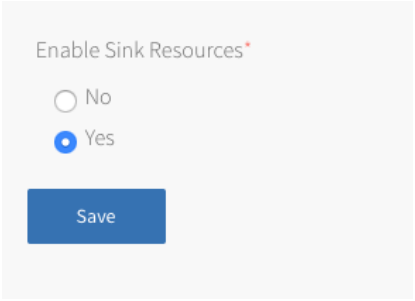
1. Click **Logging**.
2. To enable syslog forwarding for BOSH-deployed VM logs, select **Yes**.



3. Under **Address**, enter the destination syslog endpoint.
4. Under **Port**, enter the destination syslog port.
5. Select a transport protocol for log forwarding.
6. (Optional) Pivotal strongly recommends that you enable TLS encryption when forwarding logs as they may contain sensitive information. For example, these logs may contain cloud provider credentials. To enable TLS, perform the following steps:
    a. Under **Permitter Peer**, provide the accepted fingerprint (SHA1) or name of remote peer. For example, `*.YOUR-LOGGING-SYSTEM.com`.
    b. Under **TLS Certificate**, provide a TLS certificate for the destination syslog endpoint.

    > **Note**: You do not need to provide a new certificate if the TLS certificate for the destination syslog endpoint is signed by a Certificate Authority (CA) in your BOSH certificate store.

7. To enable clusters to drain Kubernetes API events and pod logs to sinks using `syslog://`, select **Enable Sink Resources**. For more information about using sink resources, see Creating Sink Resources.

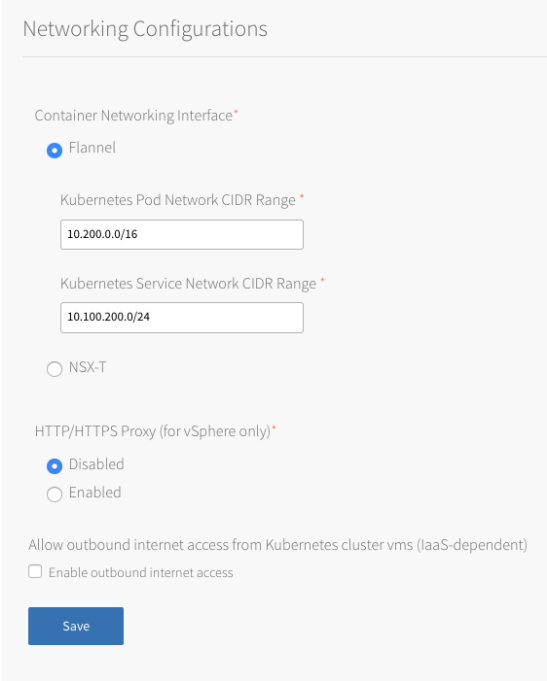


8. Click **Save**.

Networking

To configure networking, do the following:

1. Click **Networking**.

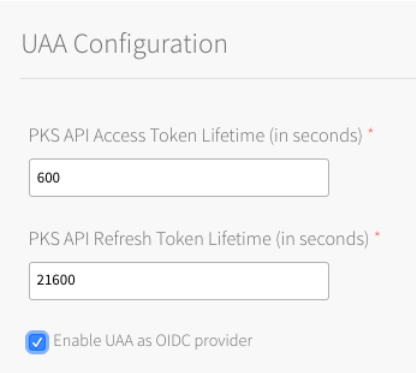


2. Under **Container Networking Interface**, select **Flannel**.
3. (Optional) Enter values for **Kubernetes Pod Network CIDR Range** and **Kubernetes Service Network CIDR Range**.
    - Ensure that the CIDR ranges do not overlap and have sufficient space for your deployed services.
    - Ensure that the CIDR range for the **Kubernetes Pod Network CIDR Range** is large enough to accommodate the expected maximum number of pods.
4. (Optional) If you do not use a NAT instance, select **Allow outbound internet access from Kubernetes cluster vms (IaaS-dependent)**. Enabling this functionality assigns external IP addresses to VMs in clusters.
5. Click **Save**.

UAA

To configure the UAA server, do the following:

1. Click **UAA**.
2. Under **PKS API Access Token Lifetime**, enter a time in seconds for the PKS API access token lifetime.



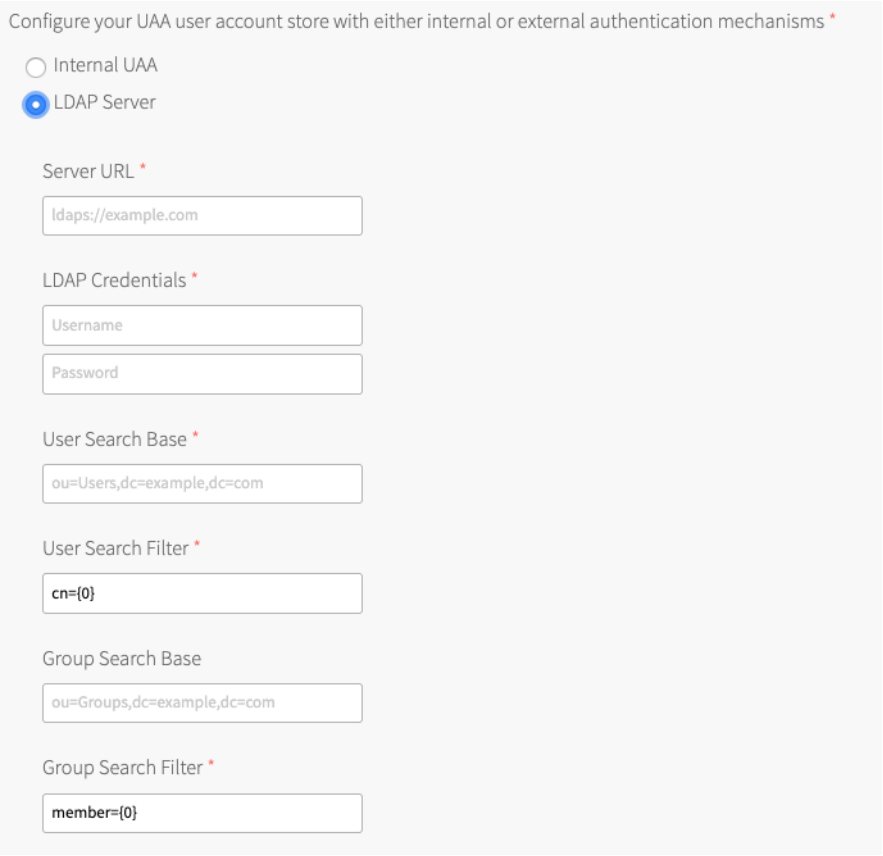
3. Under **PKS API Refresh Token Lifetime**, enter a time in seconds for the PKS API refresh token lifetime.
4. Select one of the following options:
    - To use an internal user account store for UAA, select **Internal UAA**. Click **Save** and continue to (Optional) Monitoring.
    - To use an external user account store for UAA, select **LDAP Server** and continue to Configure LDAP as an Identity Provider.

> **Note**: Selecting **LDAP Server** allows admin users to give cluster access to groups of users. For more information about performing this procedure, see Grant Cluster Access to a Group in *Managing Users in PKS with UAA*.

Configure LDAP as an Identity Provider

To integrate UAA with one or more LDAP servers, configure PKS with your LDAP endpoint information as follows:

1. Under **UAA**, select **LDAP Server**.



2. For **Server URL**, enter the URLs that point to your LDAP server. If you have multiple LDAP servers, separate their URLs with spaces. Each URL must include one of the following protocols:
    - `ldap://` : Use this protocol if your LDAP server uses an unencrypted connection.

- `ldaps://`: Use this protocol if your LDAP server uses SSL for an encrypted connection. To support an encrypted connection, the LDAP server must hold a trusted certificate or you must import a trusted certificate to the JVM truststore.

3. For **LDAP Credentials**, enter the LDAP Distinguished Name (DN) and password for binding to the LDAP server. For example, `cn=administrator,ou=Users,dc=example,dc=com`. If the bind user belongs to a different search base, you must use the full DN.

   Note: We recommend that you provide LDAP credentials that grant read-only permissions on the LDAP search base and the LDAP group search base.

4. For **User Search Base**, enter the location in the LDAP directory tree where LDAP user search begins. The LDAP search base typically matches your domain name.

   For example, a domain named `cloud.example.com` may use `ou=Users,dc=example,dc=com` as its LDAP user search base.

5. For **User Search Filter**, enter a string to use for LDAP user search criteria. The search criteria allows LDAP to perform more effective and efficient searches. For example, the standard LDAP search filter `cn=Smith` returns all objects with a common name equal to `Smith`.

   In the LDAP search filter string that you use to configure PKS, use `{0}` instead of the username. For example, use `cn={0}` to return all LDAP objects with the same common name as the username.

   In addition to `cn`, other common attributes are `mail`, `uid` and, in the case of Active Directory, `sAMAccountName`.

   Note: For information about testing and troubleshooting your LDAP search filters, see Configuring LDAP Integration with Pivot Cloud Foundry.

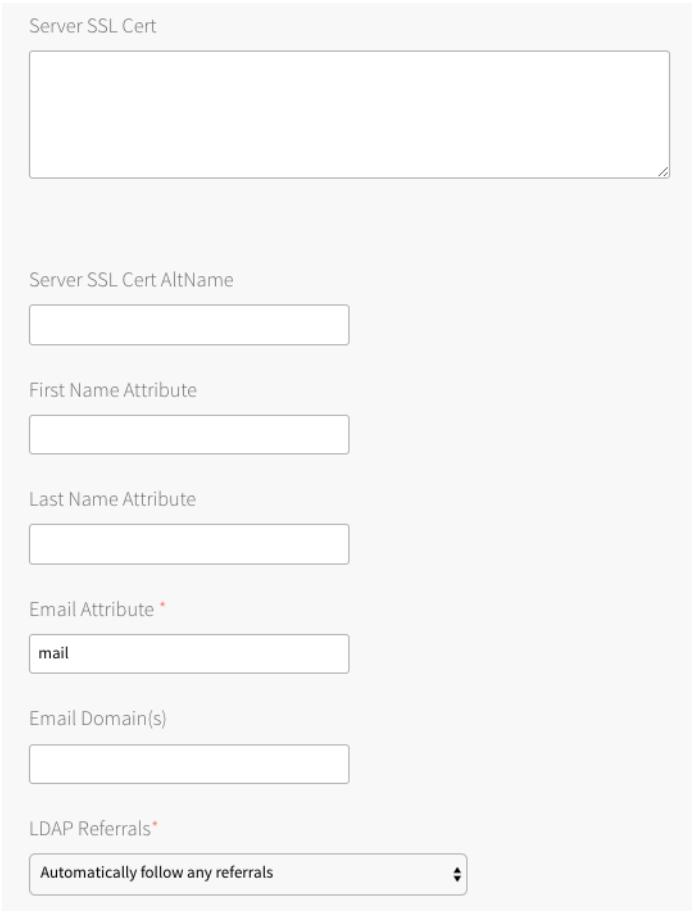
6. For **Group Search Base**, enter the location in the LDAP directory tree where the LDAP group search begins.

   For example, a domain named `cloud.example.com` may use `ou=Groups,dc=example,dc=com` as its LDAP group search base.

   Follow the instructions in the Grant PKS Access to an External LDAP Group section of *Managing Users in PKS with UAA* to map the groups under this search base to roles in PKS.

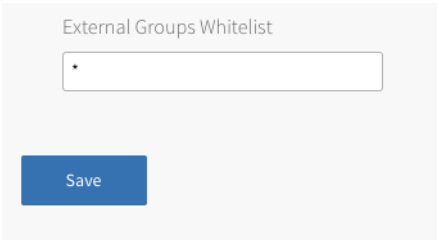
7. For **Group Search Filter**, enter a string that defines LDAP group search criteria. The standard value is `member={0}`.

8. For **Server SSL Cert**, paste in the root certificate from your CA certificate or your self-signed certificate.



9. For **Server SSL Cert AltName**, do one of the following:
    - If you are using `ldaps://` with a self-signed certificate, enter a Subject Alternative Name (SAN) for your certificate.
    - If you are not using `ldaps://` with a self-signed certificate, leave this field blank.
10. For **First Name Attribute**, enter the attribute name in your LDAP directory that contains user first names. For example, `cn`.
11. For **Last Name Attribute**, enter the attribute name in your LDAP directory that contains user last names. For example, `sn`.
12. For **Email Attribute**, enter the attribute name in your LDAP directory that contains user email addresses. For example, `mail`.
13. For **Email Domain(s)**, enter a comma-separated list of the email domains for external users who can receive invitations to Apps Manager.
14. For **LDAP Referrals**, choose how UAA handles LDAP server referrals to other user stores. UAA can follow the external referrals, ignore them without returning errors, or generate an error for each external referral and abort the authentication.
15. For **External Groups Whitelist**, enter a comma-separated list of group patterns which need to be populated in the user's `id_token`. For further information on accepted patterns see the description of the `config.externalGroupsWhitelist` in the OAuth/OIDC Identity Provider Documentation.

    Note: When sent as a Bearer token in the Authentication header, wide pattern queries for users who are members of multiple groups, can cause the size of the `id_token` to extend beyond what is supported by web servers.

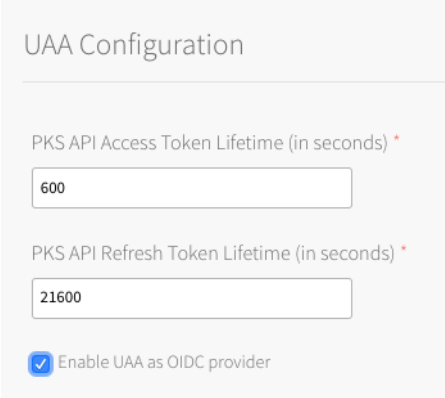


16. Click **Save**.

(Optional) Configure OpenID Connect

You can use OpenID Connect (OIDC) to instruct Kubernetes to verify end-user identities based on authentication performed by an authorization server, such as UAA.

To configure PKS to use OIDC, select **Enable UAA as OIDC provider**. With OIDC enabled, Admin Users can grant cluster-wide access to Kubernetes end users.



For more information about configuring OIDC, see the table below:

Option	Description
OIDC disabled	If you do not enable OIDC, Kubernetes authenticates users against its internal user management system.
OIDC enabled	If you enable OIDC, Kubernetes uses the authentication mechanism that you selected in UAA as follows: • If you selected **Internal UAA**, Kubernetes authenticates users against the internal UAA authentication mechanism. • If you selected **LDAP Server**, Kubernetes authenticates users against the LDAP server.

For additional information about getting credentials with OIDC configured, see Retrieve Cluster Credentials in *Retrieving Cluster Credentials and Configuration*.

Note: When you enable OIDC, existing PKS-provisioned Kubernetes clusters are upgraded to use OIDC. This invalidates your kubeconfig files. You must regenerate the files for all clusters.

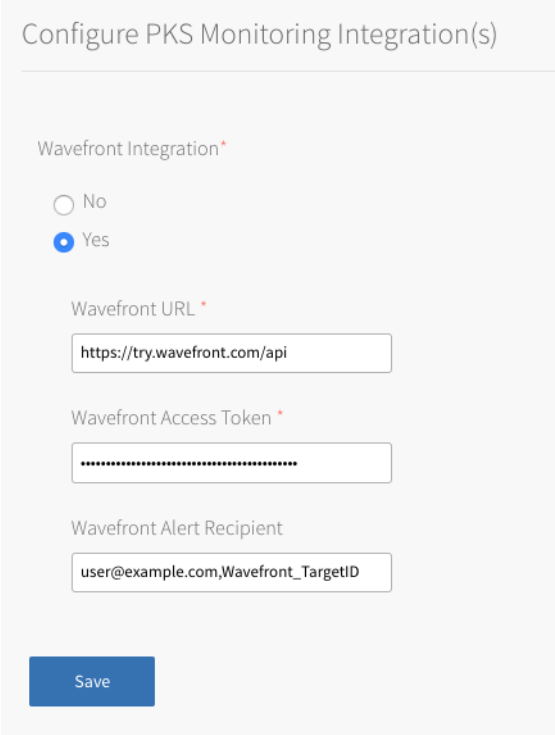
(Optional) Monitoring

You can monitor Kubernetes clusters and pods metrics externally using the integration with Wavefront by VMware.

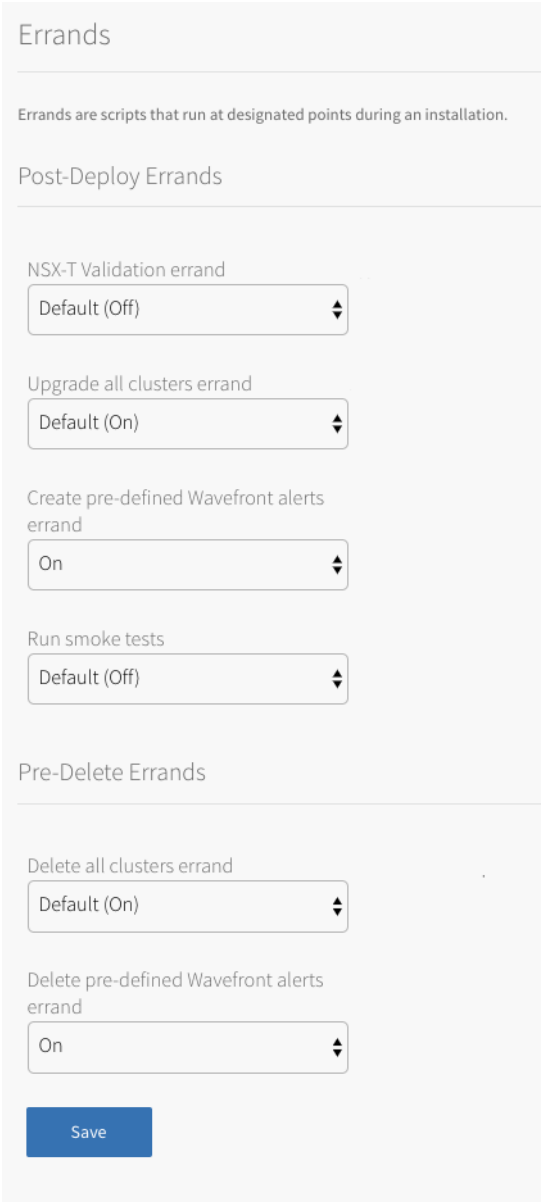
Note: Before you configure Wavefront integration, you must have an active Wavefront account and access to a Wavefront instance. You provide your Wavefront access token during configuration and enabling errands. For additional information, see Pivotal Container Service Integration Details in the Wavefront documentation.

By default, monitoring is disabled. To enable and configure Wavefront monitoring, do the following:

1. Select **Monitoring**.



2. On the **Monitoring** pane, under **Wavefront Integration**, select **Yes**.
3. Under **Wavefront URL**, enter the URL of your Wavefront subscription. For example, `https://try.wavefront.com/api`.
4. Under **Wavefront Access Token**, enter the API token for your Wavefront subscription.
5. To configure Wavefront to send alerts by email, enter email addresses or Wavefront Target IDs separated by commas under **Wavefront Alert Recipient**. For example, `user@example.com,Wavefront_TargetID`. To create alerts, you must enable errands.
6. Select **Errands**.
7. On the **Errands** pane, enable **Create pre-defined Wavefront alerts errand**.



8. Enable **Delete pre-defined Wavefront alerts errand**.
9. Click **Save**. Your settings apply to any clusters created after you have saved these configuration settings and clicked **Apply Changes**.

> **Note**: The PKS tile does not validate your Wavefront configuration settings. To verify your setup, look for cluster and pod metrics in Wavefront.

Usage Data

VMware's Customer Experience Improvement Program (CEIP) and the Pivotal Telemetry Program (Telemetry) provides VMware and Pivotal with information that enables the companies to improve their products and services, fix problems, and advise you on how best to deploy and use our products. As part of the CEIP and Telemetry, VMware and Pivotal collect technical information about your organization's use of the Pivotal Container Service (PKS) on a regular basis. Since PKS is jointly developed and sold by VMware and Pivotal, we will share this information with one another. Information collected under CEIP or Telemetry does not personally identify any individual.

Regardless of your selection in the **Usage Data** pane, a small amount of data is sent from Cloud Foundry Container Runtime (CFCR) to the PKS tile. However, that data is not shared externally.

To configure the **Usage Data** pane, perform the following steps:

1. Select the **Usage Data** side-tab.
2. Read the Usage Data description.

3. Make your selection.

    a. To join the program, select **Yes, I want to join the CEIP and Telemetry Program for PKS**.
    b. To decline joining the program, select **No, I do not want to join the CEIP and Telemetry Program for PKS**.

4. Click **Save**.

Note: If you join the CEIP and Telemetry Program for PKS, open your firewall to allow outgoing access to `https://vcsa.vmware.com/ph-prd` on port `443`.

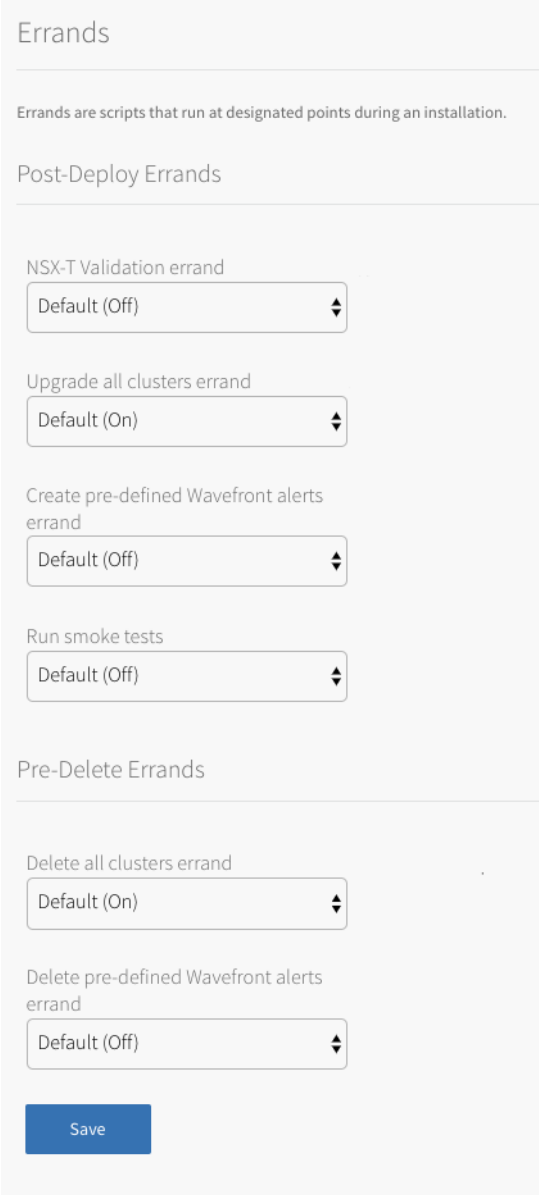
Errands

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

To configure when post-deploy and pre-delete errands for PKS are run, make a selection in the dropdown next to the errand.

We recommend that you set the **Run smoke tests** errand to **On**. The errand uses the PKS Command Line Interface (PKS CLI) to create a Kubernetes cluster and then delete it. If the creation or deletion fails, the errand fails and the installation of the PKS tile is aborted.

For the other errands, we recommend that you leave the default settings.



For more information about errands and their configuration state, see Managing Errands in Ops Manager.

WARNING: Because PKS uses floating stemcells, updating the PKS tile with a new stemcell triggers the rolling of every VM in each cluster. Also, updating other product tiles in your deployment with a new stemcell causes the PKS tile to roll VMs. This rolling is enabled by the **Upgrade all clusters errand**. We recommend that you keep this errand turned on because automatic rolling of VMs ensures that all deployed cluster VMs are patched. However, automatic rolling can cause downtime in your deployment.

If you are upgrading PKS, you must enable the **Upgrade All Clusters** errand.

Resource Config

To modify the resource usage of PKS and specify your PKS API load balancer, follow the steps below:

1. Select **Resource Config**.
2. In the **Load Balancers** column, enter all values of `pks_api_target_groups` from the Terraform output, prefixed with `alb:`.

   ```
   alb:ENV-pks-tg-9021,alb:ENV-pks-tg-8443
   ```

   Where `ENV` matches the `env_name` that you defined when you set up Terraform. For example: `alb:pcf-pks-tg-9021,alb:pcf-pks-tg-8443`

   Note: After you click **Apply Changes** for the first time, BOSH assigns the PKS VM an IP address. BOSH uses the name you provide in the **Load Balancers** column to locate your load balancer, and then connect the load balancer to the PKS VM using its new IP address.

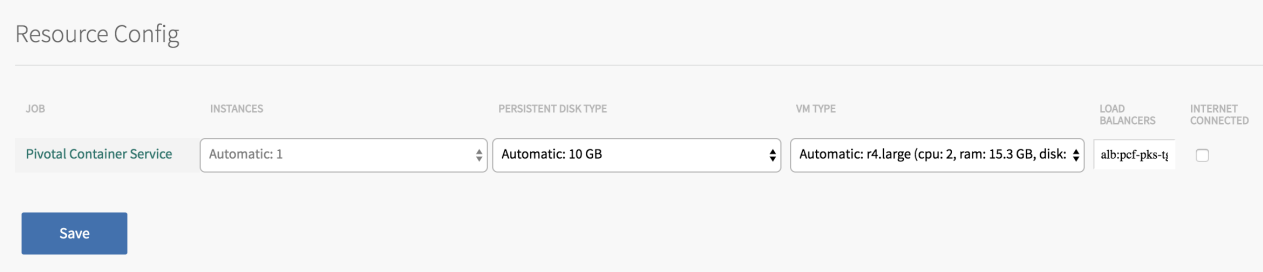
VMs used by **Pivotal Container Service** jobs must meet the following minimum requirements:

CPU	Memory	Disk
2	8 GB	29 GB

Note: If you experience timeouts or slowness when interacting with the PKS API, select a **VM Type** with greater CPU and memory resources.

To confirm you are deploying **Pivotal Container Service** job VMs meeting the minimum requirements, perform the following steps:

1. Select a **VM Type** with CPU, memory and disk resources either matching or exceeding the minimum **Pivotal Container Service** job VM requirements.

2. Select **Save**.

Step 3: Apply Changes

1. Return to the Ops Manager Installation Dashboard.
2. Click **Review Pending Changes**. Select the product that you intend to deploy and review the changes. For more information, see Reviewing Pending Product Changes.
3. Click **Apply Changes**.

Step 4: Retrieve the PKS API Endpoint

You must share the PKS API endpoint to allow your organization to use the API to create, update, and delete clusters. For more information, see Creating

Clusters.

To retrieve the PKS API endpoint, do the following:

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the Pivotal Container Service tile.
3. Click the **Status** tab and locate the **Pivotal Container Service** job. The IP address of the Pivotal Container Service job is the PKS API endpoint.

Step 5: Install the PKS and Kubernetes CLIs

The PKS and Kubernetes CLIs help you interact with your PKS-provisioned Kubernetes clusters and Kubernetes workloads. To install the CLIs, follow the instructions below:

- Installing the PKS CLI
- Installing the Kubernetes CLI

Step 6: Configure PKS API Access

Follow the procedures in Configuring PKS API Access.

Step 7: Configure Authentication for PKS

Configure authentication for PKS using User Account and Authentication (UAA). For information, see Managing Users in PKS with UAA.

Next Steps

After installing PKS on AWS, you might want to do one or more of the following:

- Create a load balancer for your PKS clusters. For more information, see Creating and Configuring an AWS Load Balancer for PKS Clusters.
- Create your first PKS cluster. For more information, see Creating Clusters.

Please send any feedback you have to pks-feedback@pivotal.io.

Azure

This topic outlines the steps for installing Pivotal Container Service (PKS) on Microsoft Azure. See the following sections:

Note: The topics below provide the Terraform procedures for deploying Ops Manager on Azure, not the manual procedures. Using the Terraform procedures is the only currently supported path for deploying Ops Manager on Azure for use with PKS.

- Azure Prerequisites and Resource Requirements
- Deploying Ops Manager 2.4 on Azure Using Terraform:
  - Preparing to Deploy Ops Manager on Azure Using Terraform
  - Deploying Ops Manager to Azure Using Terraform
  - Configuring BOSH Director on Azure Using Terraform
- Deploying Ops Manager 2.3 on Azure Using Terraform:
  - Preparing to Deploy Ops Manager on Azure Using Terraform
  - Deploying Ops Manager to Azure Using Terraform
  - Configuring BOSH Director on Azure Using Terraform
- Creating Managed Identities in Azure for PKS
- Installing PKS on Azure
- Configuring an Azure Load Balancer for the PKS API

Please send any feedback you have to pks-feedback@pivotal.io.

Azure Prerequisites and Resource Requirements

Page last updated:

This topic describes the prerequisites and resource requirements for installing Pivotal Container Service (PKS) on Microsoft Azure.

Prerequisites

Before you install PKS, you must satisfy Azure subscription requirements and install one of the following:

- Ops Manager v2.4.3 and earlier in the v2.4 version line
- Ops Manager v2.3.9 and earlier in the v2.3 version line

Note: You use Ops Manager to install and configure PKS. Each version of Ops Manager supports multiple versions of PKS. To confirm that your Ops Manager version supports the version of PKS that you install, see PKS Release Notes.

Subscription Requirements

For PKS and Kubernetes services to run correctly, you must have at least a `standard` subscription tier.

Install and Configure Ops Manager

To install an Ops Manager version that is compatible with the PKS version you intend to use, follow the instructions in the corresponding version of the Ops Manager documentation.

Note: The topics below provide the Terraform procedures for deploying Ops Manager on Azure, not the manual procedures. The Terraform procedures are the currently supported path for deploying Ops Manager on Azure for use with PKS.

Version	
Ops Manager v2.3	• Preparing to Deploy PCF on Azure Using Terraform • Configuring BOSH Director on Azure Using Terraform
Ops Manager v2.4	• Preparing to Deploy PCF on Azure Using Terraform • Configuring BOSH Director on Azure Using Terraform

Resource Requirements

Installing Ops Manager and PKS requires the following virtual machines (VMs):

VM	CPU	RAM	Storage
Pivotal Container Service	2	8 GB	16 GB ^*
Pivotal Ops Manager	1	8 GB	120 GB
BOSH Director	2	8 GB	16 GB

Storage Requirements for Large Numbers of Pods

If you expect the cluster workload to run a large number of pods continuously, then increase the size of persistent disk storage allocated to the the Pivotal Container Service VM as follows:

Number of Pods	Storage (Persistent Disk) Requirement ^*
1,000 pods	20 GB

Number of Pods	Storage (Persistent Disk) Requirement ^*
5,000 pods	100 GB
10,000 pods	200 GB
50,000 pods	1,000 GB

Kubernetes Cluster Resources

Each Kubernetes cluster provisioned through PKS deploys the VMs listed below. If you deploy more than one Kubernetes cluster, you must scale your allocated resources appropriately.

VM Name	Number	CPU Cores	RAM	Ephemeral Disk	Persistent Disk
master	1	2	4 GB	32 GB	5 GB
worker	1	2	4 GB	32 GB	50 GB

Please send any feedback you have to pks-feedback@pivotal.io.

Creating Managed Identities in Azure for PKS

Page last updated:

This topic describes how to create managed identities for Pivotal Container Service (PKS) on Azure.

In order for Kubernetes to create load balancers and attach persistent disks to pods, you must create managed identities with sufficient permissions.

You need separate managed identities for the Kubernetes cluster master and worker node VMs. Pivotal recommends configuring each service account with the least permissive privileges and unique credentials.

Retrieve Your Subscription ID and Resource Group

To perform the procedures in this topic, you must retrieve your subscription ID and the name of your PKS resource group.

You entered your subscription ID into the `terraform.tfvars` file in Step 1: Download and Edit the Terraform Variables File of *Deploying Ops Manager on Azure*.

The name of your PKS resource group is exported from Terraform as the output `pcf_resource_group_name`.

To retrieve your subscription ID and the name of your PKS resource group, you must have access to the output from when you ran `terraform apply` to create resources for the PKS deployment in Deploying Ops Manager to Azure Using Terraform. You can view this output at any time by running `terraform output`.

Create the Master Node Managed Identity

Perform the following steps to create the managed identity for the master nodes:

1. Create a role definition using the following template, replacing `SUBSCRIPTION_ID` and `RESOURCE_GROUP` with your subscription ID and the name of your PKS resource group. For more information about custom roles in Azure, see Custom Roles in Azure in the Azure documentation.

```
{
  "Name": "PKS master",
  "IsCustom": true,
  "Description": "Permissions for PKS master",
  "Actions": [
    "Microsoft.Network/*",
    "Microsoft.Compute/disks/*",
    "Microsoft.Compute/virtualMachines/write",
    "Microsoft.Compute/virtualMachines/read",
    "Microsoft.Storage/storageAccounts/*"
  ],
  "NotActions": [

  ],
  "DataActions": [

  ],
  "NotDataActions": [

  ],
  "AssignableScopes": [
    "/subscriptions/SUBSCRIPTION-ID/resourceGroups/RESOURCE-GROUP"
  ]
}
```

2. Save your template as `pks_master_role.json`.

3. To log in, run the following command with the Azure CLI:

```
az login
```

   To authenticate, navigate to the URL in the output, enter the provided code, and click your account.

4. Create the role in Azure by running the following command from the directory with `pks_master_role.json`:

```
az role definition create --role-definition pks_master_role.json
```

5. Create a managed identity by running the following command:

```
az identity create -g RESOURCE_GROUP -n pks-master
```

Where `RESOURCE_GROUP` is the name of your PKS resource group.

For more information about managed identities, see Create a user-assigned managed identity in the Azure documentation.

6. Assign managed identity access to the PKS resource group by performing the following steps:

    a. Navigate to the Azure Portal and log in.
    b. Open the PKS resource group.
    c. Click **Access control (IAM)** on the left panel.
    d. Click **Add role assignment**.
    e. On the **Add role assignment** page, enter the following configurations:

        i. For **Assign access to**, select **User Assigned Managed Identity**.
        ii. For **Role**, select **PKS master**.
        iii. For **Select**, select the **pks-master** identity created above.

Note: The **PKS master** custom role created above is less permissive than the built-in roles provided by Azure. However, if you want to use the built-in roles instead of the recommended custom role, you can select the following three built-in roles in Azure: **Storage Account Contributor**, **Network Contributor**, and **Virtual Machine Contributor**.

Create the Worker Node Managed Identity

Perform the following steps to create the managed identity for the worker nodes:

1. Create a role definition using the following template, replacing `SUBSCRIPTION-ID` and `RESOURCE-GROUP` with your subscription ID and the name of your PKS resource group:

```
{
  "Name": "PKS worker",
  "IsCustom": true,
  "Description": "Permissions for PKS worker",
  "Actions": [
    "Microsoft.Storage/storageAccounts/*"
  ],
  "NotActions": [

  ],
  "DataActions": [

  ],
  "NotDataActions": [

  ],
  "AssignableScopes": [
   "/subscriptions/SUBSCRIPTION-ID/resourceGroups/RESOURCE-GROUP"
  ]
}
```

2. Save your template as `pks_worker_role.json`.

3. Create the role in Azure by running the following command from the directory with `pks_worker_role.json`:

```
az role definition create --role-definition pks_worker_role.json
```

4. Create a managed identity by running the following command:

```
az identity create -g RESOURCE_GROUP -n pks-worker
```

Where `RESOURCE_GROUP` is the name of your PKS resource group.

5. Assign managed identity access to the PKS resource group by performing the following steps:

    a. Navigate to the Azure Portal and log in.

b. Open the PKS resource group.
c. Click **Access control (IAM)** on the left panel.
d. Click **Add role assignment**.
e. On the **Add role assignment** page, enter the following configurations:

    i. For **Assign access to**, select **User Assigned Managed Identity**.
    ii. For **Role**, select **PKS worker**.
    iii. For **Select**, select the **pks-worker** identity created above.

Note: The **PKS worker** custom role created above is less permissive than the built-in roles provided by Azure. However, if you want to use the built-in roles instead of the recommended custom role, you can select the **Storage Account Contributor** built-in role in Azure.

After you create managed identities for both the master and worker nodes, follow the procedures in Installing PKS on Azure.

Please send any feedback you have to pks-feedback@pivotal.io.

Installing PKS on Azure

Page last updated:

This topic describes how to install and configure Pivotal Container Service (PKS) on Azure.

Prerequisites

Before performing the procedures in this topic, you must have deployed and configured Ops Manager. For more information, see Azure Prerequisites and Resource Requirements.

If you use an instance of Ops Manager that you configured previously to install other runtimes, perform the following steps before you install PKS:

1. Navigate to Ops Manager.
2. Open the **Director Config** pane.
3. Select the **Enable Post Deploy Scripts** checkbox.
4. Click the **Installation Dashboard** link to return to the Installation Dashboard.
5. Click **Review Pending Changes**. Select all products you intend to deploy and review the changes. For more information, see Reviewing Pending Product Changes.
6. Click **Apply Changes**.

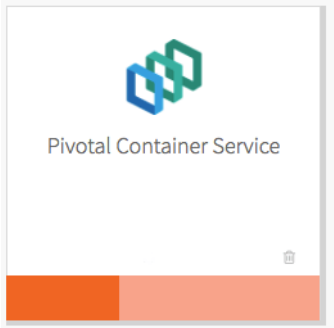
Step 1: Install PKS

To install PKS, do the following:

1. Download the product file from Pivotal Network.
2. Navigate to `https://YOUR-OPS-MANAGER-FQDN/` in a browser to log in to the Ops Manager Installation Dashboard.
3. Click **Import a Product** to upload the product file.
4. Under **Pivotal Container Service** in the left column, click the plus sign to add this product to your staging area.

Step 2: Configure PKS

Click the orange **Pivotal Container Service** tile to start the configuration process.

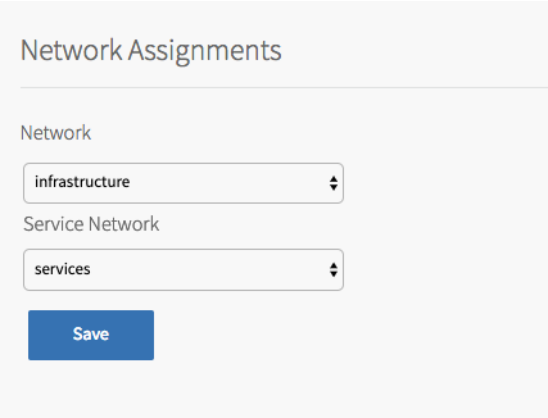


WARNING: When you configure the PKS tile, do not use spaces in any field entries. This includes spaces between characters as well as leading and trailing spaces. If you use a space in any field entry, the deployment of PKS fails.

Assign Networks

Perform the following steps:

1. Click **Assign Networks**.

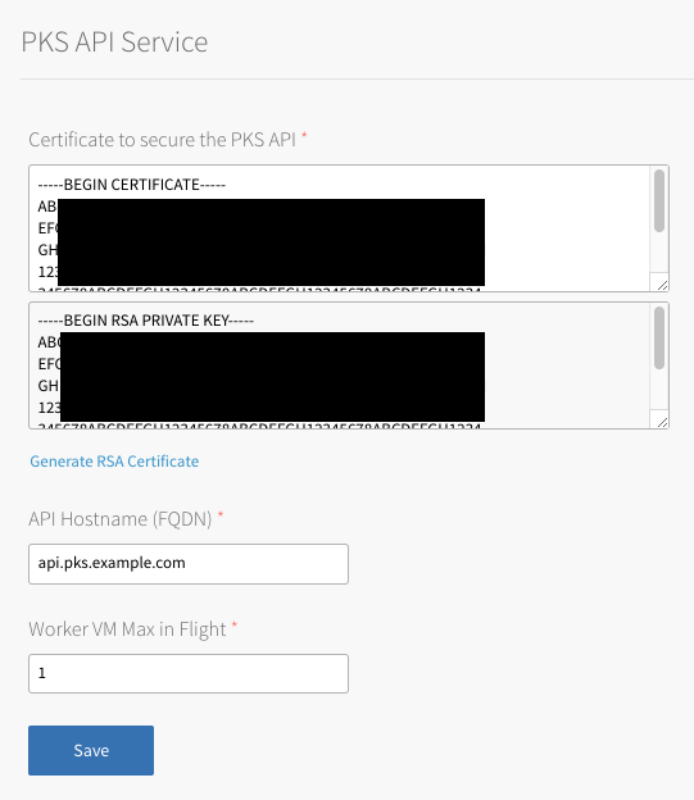


2. Under **Network**, select the PKS subnet that you created for the PKS API VM. For example, `infrastructure`.
3. Under **Service Network**, select the services subnet that you created for Kubernetes cluster VMs. For example, `services`.
4. Click **Save**.

PKS API

Perform the following steps:

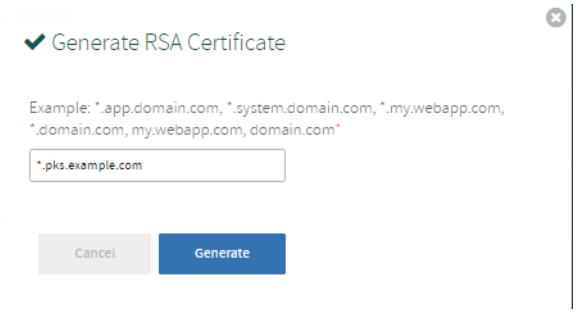
1. Click **PKS API**.
2. Under **Certificate to secure the PKS API**, provide your own certificate and private key pair.



The certificate that you supply should cover the domain that routes to the PKS API VM with TLS termination on the ingress.

If you do not have a certificate and private key pair, PKS can generate one for you. To generate a certificate, do the following:

a. Select the **Generate RSA Certificate** link.
b. Enter the domain for your API hostname. This can be a standard FQDN or a wildcard domain.
c. Click **Generate**.



3. Under **API Hostname (FQDN)**, enter the FQDN that you registered to point to the PKS API load balancer, such as `api.pks.example.com`. To retrieve the public IP address or FQDN of the PKS API load balancer, see the `terraform.tfstate` file.

4. Under **Worker VM Max in Flight**, enter the maximum number of non-canary worker instances to create or resize in parallel within an availability zone.

   This field sets the `max_in_flight` variable, which limits how many instances of a component can start simultaneously when a cluster is created or resized. The variable defaults to `1`, which means that only one component starts at a time.

5. Click **Save**.

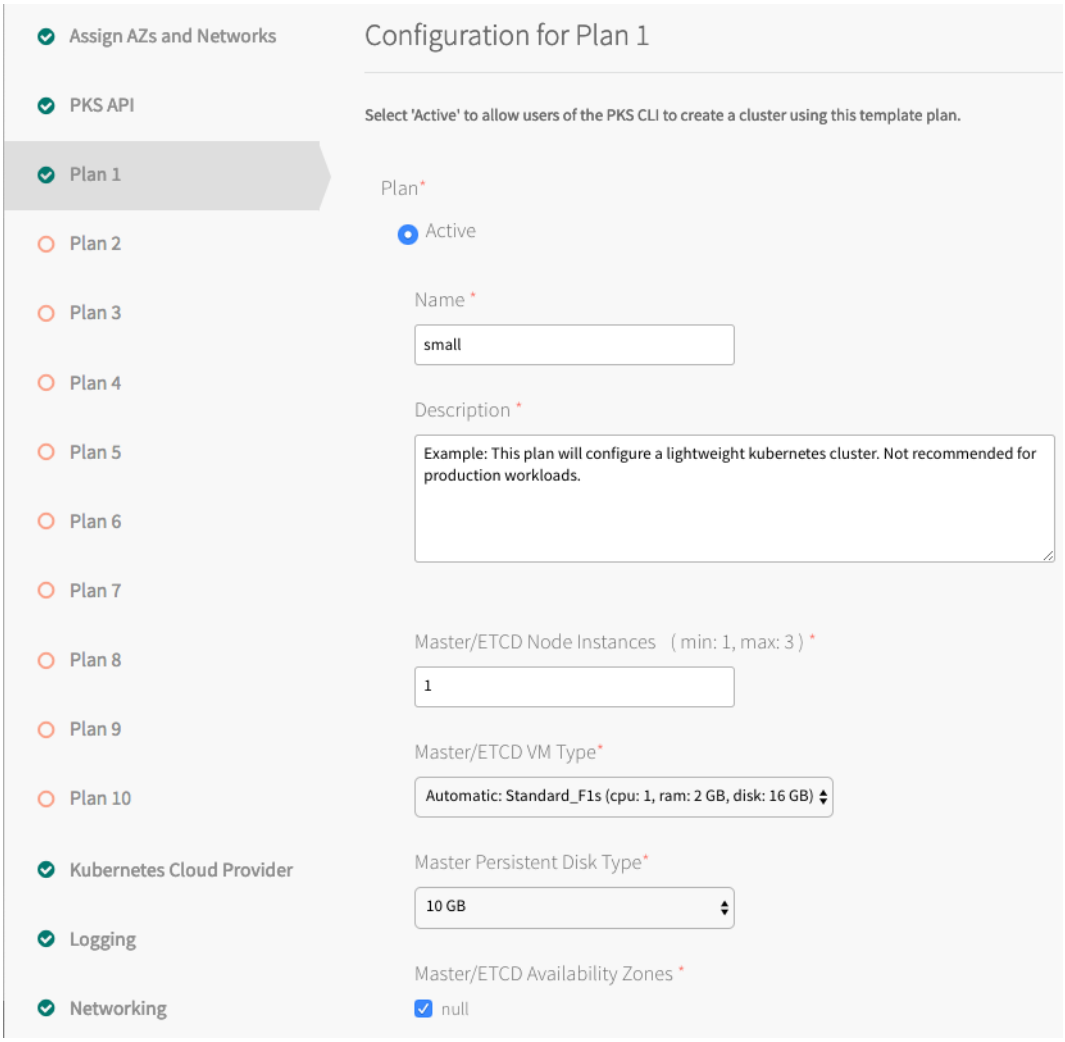
Plans

To activate a plan, perform the following steps:

1. Click the plan that you want to activate.

   Note: A plan defines a set of resource types used for deploying clusters. You can configure up to ten plans. You must configure **Plan 1**.

2. Select **Active** to activate the plan and make it available to developers deploying clusters.



3. Under **Name**, provide a unique name for the plan.

4. Under **Description**, edit the description as needed. The plan description appears in the Services Marketplace, which developers can access by using PKS CLI.

5. Under **Master/ETCD Node Instances**, select the default number of Kubernetes master/etcd nodes to provision for each cluster. You can enter either `1` or `3`.

   Note: If you deploy a cluster with multiple master/etcd node VMs, confirm that you have sufficient hardware to handle the increased load on disk write and network traffic. For more information, see Hardware recommendations in the etcd documentation.

   In addition to meeting the hardware requirements for a multi-master cluster, we recommend configuring monitoring for etcd to monitor disk latency, network latency, and other indicators for the health of the cluster. For more information, see Monitoring Master/etcd Node VMs.

   WARNING: To change the number of master/etcd nodes for a plan, you must ensure that no existing clusters use the plan. PKS does not support changing the number of master/etcd nodes for plans with existing clusters.

6. Under **Master/ETCD VM Type**, select the type of VM to use for Kubernetes master/etcd nodes. For more information, including master node VM customization options, see the Master Node VM Size section of *VM Sizing for PKS Clusters*.

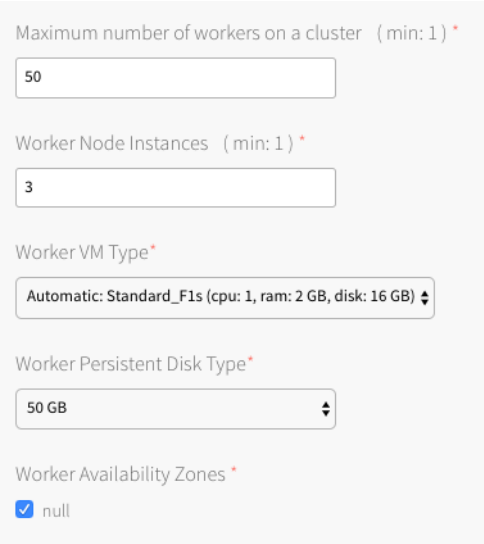
7. Under **Master Persistent Disk Type**, select the size of the persistent disk for the Kubernetes master node VM.

8. Under **Master/ETCD Availability Zones**, select **null**.

   Note: Ops Manager on Azure does not support availability zones. By default, BOSH deploys VMs in Azure Availability Sets.

9. Under **Maximum number of workers on a cluster**, set the maximum number of Kubernetes worker node VMs that PKS can deploy for each cluster.

Enter any whole number in this field.



10. Under **Worker Node Instances**, select the default number of Kubernetes worker nodes to provision for each cluster.

    If the user creating a cluster with the PKS CLI does not specify a number of worker nodes, the cluster is deployed with the default number set in this field. This value cannot be greater than the maximum worker node value you set in the previous field. For more information about creating clusters, see Creating Clusters.

    For high availability, create clusters with a minimum of three worker nodes, or two per AZ if you intend to use PersistentVolumes (PVs). For example, if you deploy across three AZs, you should have six worker nodes. For more information about PVs, see PersistentVolumes in *Maintaining Workload Uptime*. Provisioning a minimum of three worker nodes, or two nodes per AZ is also recommended for stateless workloads.

    If you later reconfigure the plan to adjust the default number of worker nodes, the existing clusters that have been created from that plan are not automatically upgraded with the new default number of worker nodes.

11. Under **Worker VM Type**, select the type of VM to use for Kubernetes worker node VMs. For more information, including worker node VM customization options, see the Worker Node VM Number and Size section of *VM Sizing for PKS Clusters*.

    Note: If you install PKS in an NSX-T environment, we recommend that you select a **Worker VM Type** with a minimum disk size of 16 GB. The disk space provided by the default `medium` Worker VM Type is insufficient for PKS with NSX-T.

12. Under **Worker Persistent Disk Type**, select the size of the persistent disk for the Kubernetes worker node VMs.

13. Under **Worker Availability Zones**, select **null**.

    Note: Ops Manager on Azure does not support availability zones. By default, BOSH deploys VMs in Azure Availability Sets.

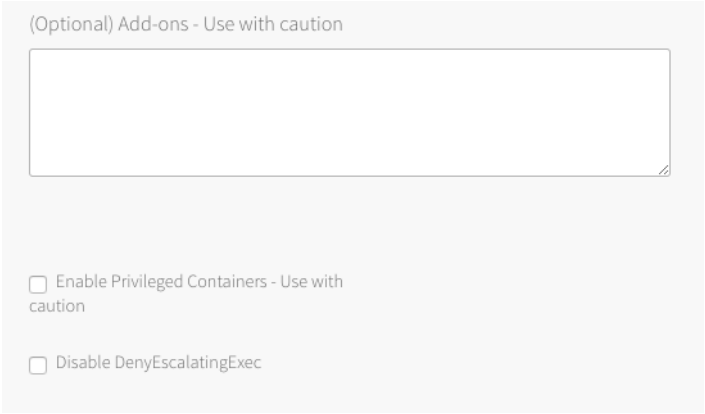
14. Under **Kubelet customization - system-reserved**, enter resource values that Kubelet can use to reserve resources for system daemons. For example, `memory=250Mi, cpu=150m`. For more information about system-reserved values, see the Kubernetes documentation.

15. Under **Kubelet customization - eviction-hard**, enter threshold limits that Kubelet can use to evict pods when they exceed the limit. Enter limits in the format `EVICTION-SIGNAL=QUANTITY`. For example, `memory.available=100Mi, nodefs.available=10%, nodefs.inodesFree=5%`. For more information about eviction thresholds, see the Kubernetes documentation.

    WARNING: Use the Kubelet customization fields with caution. If you enter values that are invalid or that exceed the limits the system supports, Kubelet might fail to start. If Kubelet fails to start, you cannot create clusters.

16. Under **Errand VM Type**, select the size of the VM that contains the errand. The smallest instance possible is sufficient, as the only errand running on this VM is the one that applies the **Default Cluster App** YAML configuration.

17. (Optional) Under **(Optional) Add-ons - Use with caution**, enter additional YAML configuration to add custom workloads to each cluster in this plan. You can specify multiple files using `---` as a separator. For more information, see Adding Custom Workloads.



18. (Optional) To allow users to create pods with privileged containers, select the **Enable Privileged Containers - Use with caution** option. For more information, see Pods in the Kubernetes documentation.
19. (Optional) To disable the admission controller, select the **Disable DenyEscalatingExec** checkbox. If you select this option, clusters in this plan can create security vulnerabilities that may impact other tiles. Use this feature with caution.
20. Click **Save**.

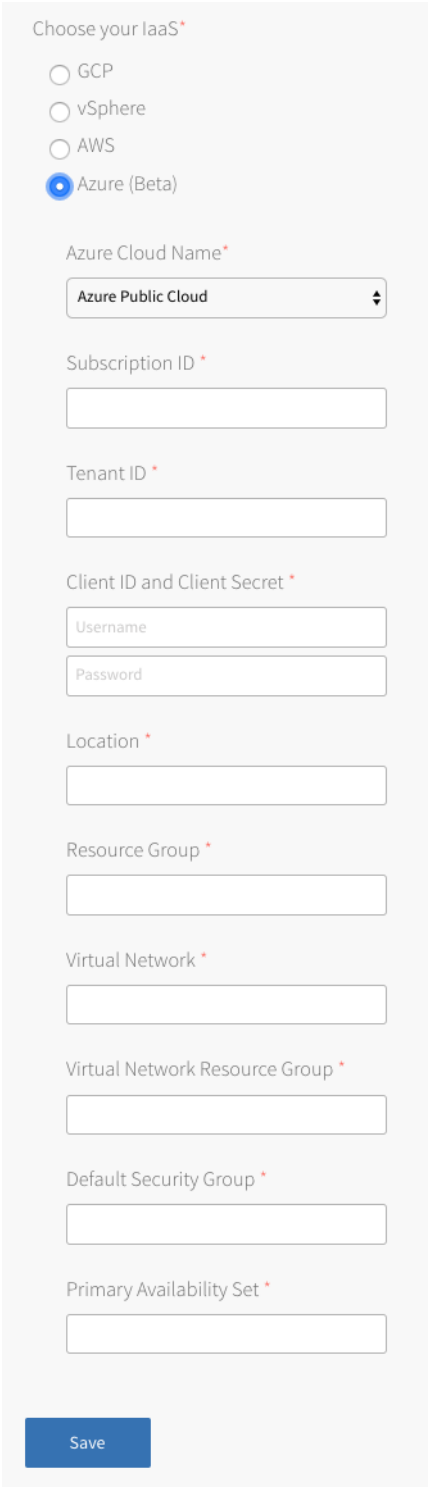
To deactivate a plan, perform the following steps:

1. Click the plan that you want to deactivate.
2. Select **Inactive**.
3. Click **Save**.

Kubernetes Cloud Provider

To configure your Kubernetes cloud provider settings, follow the procedures below:

1. Click **Kubernetes Cloud Provider**.
2. Under **Choose your IaaS**, select **Azure**.



3. Under **Azure Cloud Name**, select the identifier of your Azure environment.

4. Enter **Subscription ID**. This is the ID of the Azure subscription that the cluster is deployed in.

5. Enter **Tenant ID**. This is the Azure Active Directory (AAD) tenant ID for the subscription that the cluster is deployed in.

6. Enter **Location**. This is the location of the resource group that the cluster is deployed in.

   You set the location name in the `terraform.tfvars` file in Deploying Ops Manager to Azure Using Terraform. However, Terraform removes the spaces from this name and makes it lower-case. For example, if you entered `Central US` in the `terraform.tfvars` file, it becomes `centralus`. You must enter the converted form of the location name in the **Location** field, such as `centralus`.

7. Enter **Resource Group**. This is the name of the resource group that the cluster is deployed in.

8. Enter **Virtual Network**. This is the name of the virtual network that the cluster is deployed in.
9. Enter **Virtual Network Resource Group**. This is the name of the resource group that the virtual network is deployed in.
10. Enter **Default Security Group**. This is the name of the security group attached to the cluster's subnet.
11. Enter **Primary Availability Set**. This is the name of the availability set that will be used as the load balancer back end.

    Terraform creates this availability set and its name is `YOUR-ENVIRONMENT-NAME-pks-as`, where `YOUR-ENVIRONMENT-NAME` is the value you provided for `env_name` in the `terraform.tfvars` file. See Step 1: Download and Edit the Terraform Variables File of *Deploying Ops Manager to Azure Using Terraform* for more information. You can also find the name of the availability set by logging in to the Azure console.
12. For **Master Managed Identity**, enter `pks-master`. You created the managed identity for the master nodes in Create the Master Nodes Managed Identity in *Creating Managed Identities in Azure for PKS*.
13. For **Worker Managed Identity**, enter `pks-worker`. You created the managed identity for the worker nodes in Create the Worker Nodes Managed Identity in *Creating Managed Identities in Azure for PKS*.
14. Click **Save**.

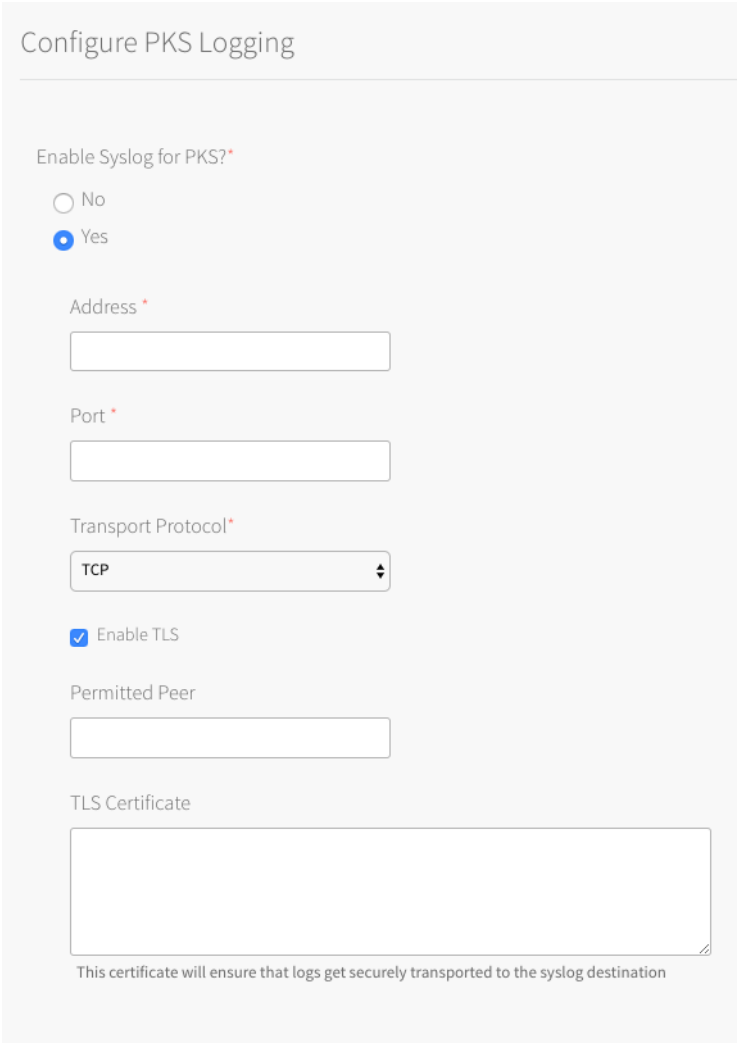
(Optional) Logging

You can designate an external syslog endpoint for forwarded BOSH-deployed VM logs.

In addition, you can enable sink resources to collect PKS cluster and namespace log messages.

To configure logging in PKS, do the following:

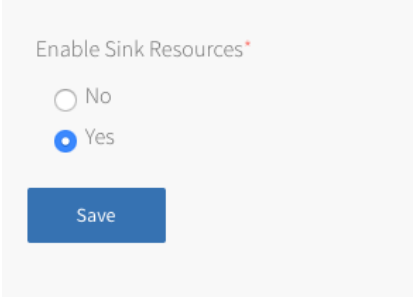
1. Click **Logging**.
2. To enable syslog forwarding for BOSH-deployed VM logs, select **Yes**.



3. Under **Address**, enter the destination syslog endpoint.
4. Under **Port**, enter the destination syslog port.
5. Select a transport protocol for log forwarding.
6. (Optional) Pivotal strongly recommends that you enable TLS encryption when forwarding logs as they may contain sensitive information. For example, these logs may contain cloud provider credentials. To enable TLS, perform the following steps:
    a. Under **Permitter Peer**, provide the accepted fingerprint (SHA1) or name of remote peer. For example, `*.YOUR-LOGGING-SYSTEM.com`.
    b. Under **TLS Certificate**, provide a TLS certificate for the destination syslog endpoint.

    > **Note:** You do not need to provide a new certificate if the TLS certificate for the destination syslog endpoint is signed by a Certificate Authority (CA) in your BOSH certificate store.

7. To enable clusters to drain Kubernetes API events and pod logs to sinks using `syslog://`, select **Enable Sink Resources**. For more information about using sink resources, see Creating Sink Resources.

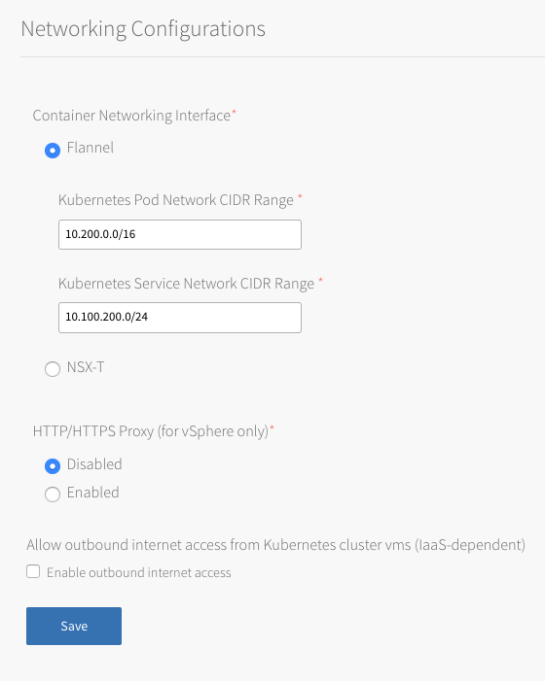


8. Click **Save**.

Networking

To configure networking, do the following:

1. Click **Networking**.

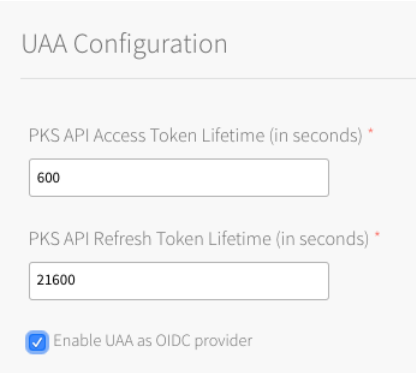


2. Under **Container Networking Interface**, select **Flannel**.
3. (Optional) Enter values for **Kubernetes Pod Network CIDR Range** and **Kubernetes Service Network CIDR Range**.
    - Ensure that the CIDR ranges do not overlap and have sufficient space for your deployed services.
    - Ensure that the CIDR range for the **Kubernetes Pod Network CIDR Range** is large enough to accommodate the expected maximum number of pods.
4. Under **Allow outbound internet access from Kubernetes cluster vms (IaaS-dependent)**, leave the **Enable outbound internet access** checkbox unselected. You must leave this checkbox unselected due to an incompatibility between the public dynamic IPs provided by BOSH and load balancers on Azure.
5. Click **Save**.

UAA

To configure the UAA server, do the following:

1. Click **UAA**.
2. Under **PKS API Access Token Lifetime**, enter a time in seconds for the PKS API access token lifetime.



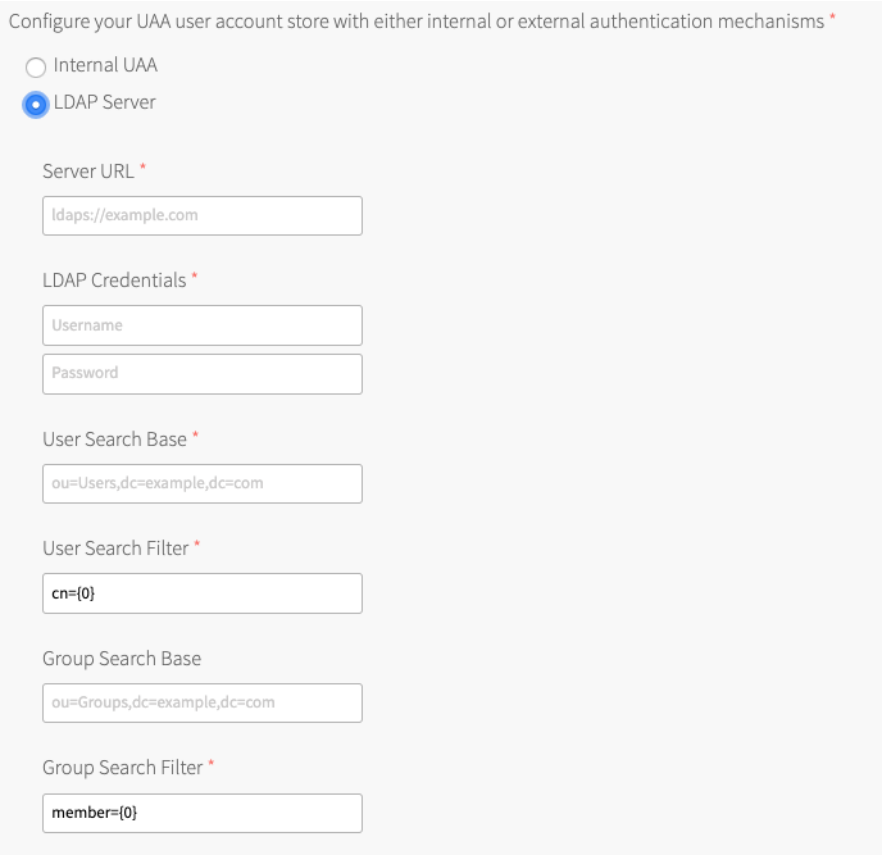
3. Under **PKS API Refresh Token Lifetime**, enter a time in seconds for the PKS API refresh token lifetime.
4. Select one of the following options:
    - To use an internal user account store for UAA, select **Internal UAA**. Click **Save** and continue to (Optional) Monitoring.
    - To use an external user account store for UAA, select **LDAP Server** and continue to Configure LDAP as an Identity Provider.

> **Note**: Selecting **LDAP Server** allows admin users to give cluster access to groups of users. For more information about performing this procedure, see Grant Cluster Access to a Group in *Managing Users in PKS with UAA*.

Configure LDAP as an Identity Provider

To integrate UAA with one or more LDAP servers, configure PKS with your LDAP endpoint information as follows:

1. Under **UAA**, select **LDAP Server**.



2. For **Server URL**, enter the URLs that point to your LDAP server. If you have multiple LDAP servers, separate their URLs with spaces. Each URL must include one of the following protocols:
    - `ldap://` : Use this protocol if your LDAP server uses an unencrypted connection.

- `ldaps://`: Use this protocol if your LDAP server uses SSL for an encrypted connection. To support an encrypted connection, the LDAP server must hold a trusted certificate or you must import a trusted certificate to the JVM truststore.

3. For **LDAP Credentials**, enter the LDAP Distinguished Name (DN) and password for binding to the LDAP server. For example, `cn=administrator,ou=Users,dc=example,dc=com`. If the bind user belongs to a different search base, you must use the full DN.

   Note: We recommend that you provide LDAP credentials that grant read-only permissions on the LDAP search base and the LDAP group search base.

4. For **User Search Base**, enter the location in the LDAP directory tree where LDAP user search begins. The LDAP search base typically matches your domain name.

   For example, a domain named `cloud.example.com` may use `ou=Users,dc=example,dc=com` as its LDAP user search base.

5. For **User Search Filter**, enter a string to use for LDAP user search criteria. The search criteria allows LDAP to perform more effective and efficient searches. For example, the standard LDAP search filter `cn=Smith` returns all objects with a common name equal to `Smith`.

   In the LDAP search filter string that you use to configure PKS, use `{0}` instead of the username. For example, use `cn={0}` to return all LDAP objects with the same common name as the username.

   In addition to `cn`, other common attributes are `mail`, `uid` and, in the case of Active Directory, `sAMAccountName`.

   Note: For information about testing and troubleshooting your LDAP search filters, see Configuring LDAP Integration with Pivot Cloud Foundry.

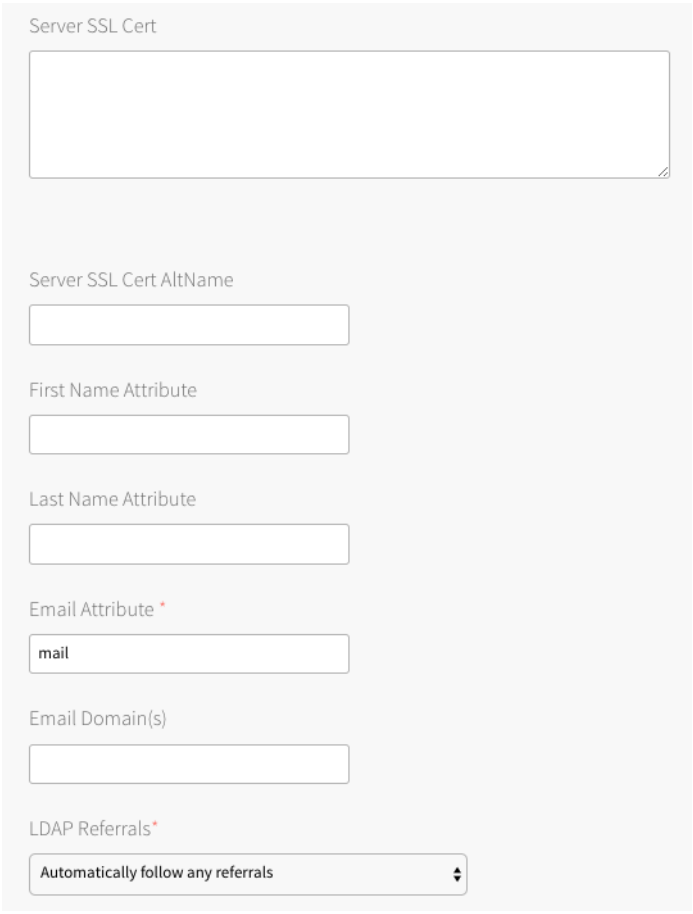
6. For **Group Search Base**, enter the location in the LDAP directory tree where the LDAP group search begins.

   For example, a domain named `cloud.example.com` may use `ou=Groups,dc=example,dc=com` as its LDAP group search base.

   Follow the instructions in the Grant PKS Access to an External LDAP Group section of *Managing Users in PKS with UAA* to map the groups under this search base to roles in PKS.

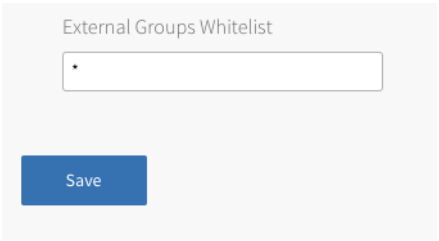
7. For **Group Search Filter**, enter a string that defines LDAP group search criteria. The standard value is `member={0}`.

8. For **Server SSL Cert**, paste in the root certificate from your CA certificate or your self-signed certificate.



9. For **Server SSL Cert AltName**, do one of the following:
    - If you are using `ldaps://` with a self-signed certificate, enter a Subject Alternative Name (SAN) for your certificate.
    - If you are not using `ldaps://` with a self-signed certificate, leave this field blank.
10. For **First Name Attribute**, enter the attribute name in your LDAP directory that contains user first names. For example, `cn`.
11. For **Last Name Attribute**, enter the attribute name in your LDAP directory that contains user last names. For example, `sn`.
12. For **Email Attribute**, enter the attribute name in your LDAP directory that contains user email addresses. For example, `mail`.
13. For **Email Domain(s)**, enter a comma-separated list of the email domains for external users who can receive invitations to Apps Manager.
14. For **LDAP Referrals**, choose how UAA handles LDAP server referrals to other user stores. UAA can follow the external referrals, ignore them without returning errors, or generate an error for each external referral and abort the authentication.
15. For **External Groups Whitelist**, enter a comma-separated list of group patterns which need to be populated in the user's `id_token`. For further information on accepted patterns see the description of the `config.externalGroupsWhitelist` in the OAuth/OIDC Identity Provider Documentation.

    Note: When sent as a Bearer token in the Authentication header, wide pattern queries for users who are members of multiple groups, can cause the size of the `id_token` to extend beyond what is supported by web servers.

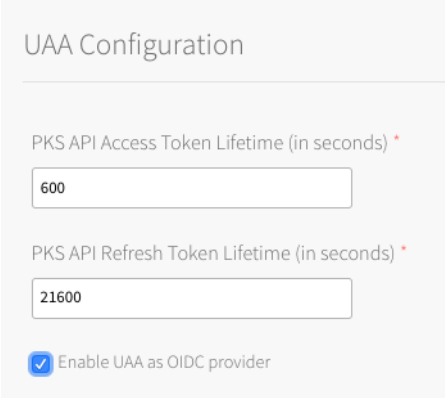


16. Click **Save**.

(Optional) Configure OpenID Connect

You can use OpenID Connect (OIDC) to instruct Kubernetes to verify end-user identities based on authentication performed by an authorization server, such as UAA.

To configure PKS to use OIDC, select **Enable UAA as OIDC provider**. With OIDC enabled, Admin Users can grant cluster-wide access to Kubernetes end users.



For more information about configuring OIDC, see the table below:

Option	Description
OIDC disabled	If you do not enable OIDC, Kubernetes authenticates users against its internal user management system.
OIDC enabled	If you enable OIDC, Kubernetes uses the authentication mechanism that you selected in UAA as follows: • If you selected **Internal UAA**, Kubernetes authenticates users against the internal UAA authentication mechanism. • If you selected **LDAP Server**, Kubernetes authenticates users against the LDAP server.

For additional information about getting credentials with OIDC configured, see Retrieve Cluster Credentials in *Retrieving Cluster Credentials and Configuration*.

Note: When you enable OIDC, existing PKS-provisioned Kubernetes clusters are upgraded to use OIDC. This invalidates your kubeconfig files. You must regenerate the files for all clusters.

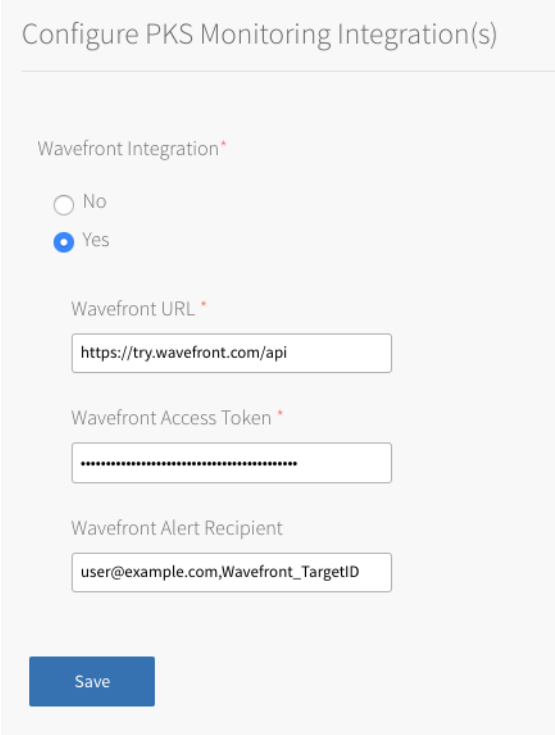
(Optional) Monitoring

You can monitor Kubernetes clusters and pods metrics externally using the integration with Wavefront by VMware.

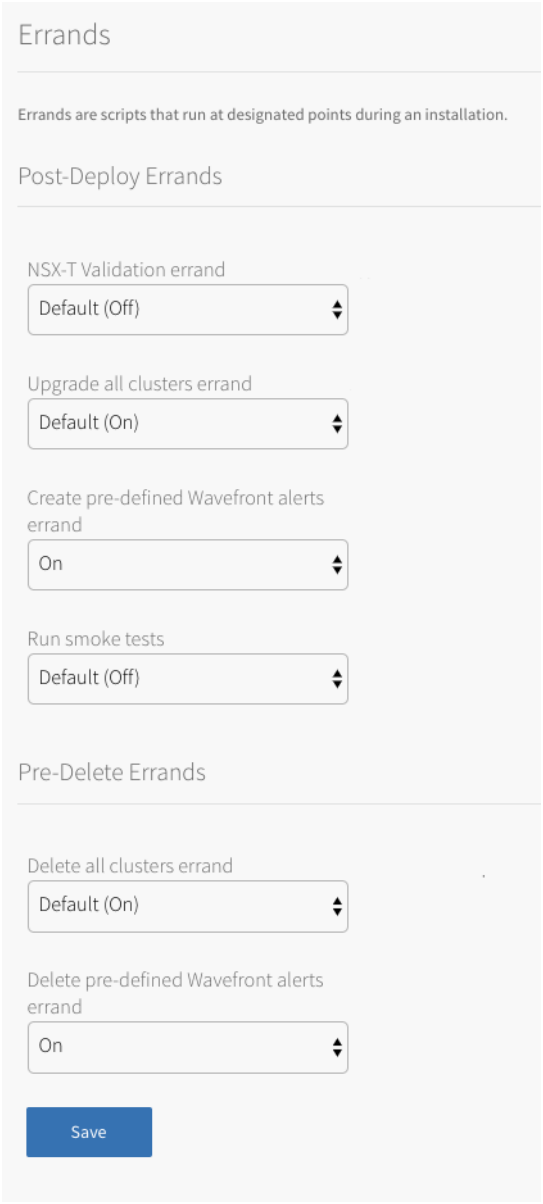
Note: Before you configure Wavefront integration, you must have an active Wavefront account and access to a Wavefront instance. You provide your Wavefront access token during configuration and enabling errands. For additional information, see Pivotal Container Service Integration Details in the Wavefront documentation.

By default, monitoring is disabled. To enable and configure Wavefront monitoring, do the following:

1. Select **Monitoring**.



2. On the **Monitoring** pane, under **Wavefront Integration**, select **Yes**.
3. Under **Wavefront URL**, enter the URL of your Wavefront subscription. For example, `https://try.wavefront.com/api`.
4. Under **Wavefront Access Token**, enter the API token for your Wavefront subscription.
5. To configure Wavefront to send alerts by email, enter email addresses or Wavefront Target IDs separated by commas under **Wavefront Alert Recipient**. For example, `user@example.com,Wavefront_TargetID`. To create alerts, you must enable errands.
6. Select **Errands**.
7. On the **Errands** pane, enable **Create pre-defined Wavefront alerts errand**.



8. Enable **Delete pre-defined Wavefront alerts errand**.
9. Click **Save**. Your settings apply to any clusters created after you have saved these configuration settings and clicked **Apply Changes**.

> **Note**: The PKS tile does not validate your Wavefront configuration settings. To verify your setup, look for cluster and pod metrics in Wavefront.

Usage Data

VMware's Customer Experience Improvement Program (CEIP) and the Pivotal Telemetry Program (Telemetry) provides VMware and Pivotal with information that enables the companies to improve their products and services, fix problems, and advise you on how best to deploy and use our products. As part of the CEIP and Telemetry, VMware and Pivotal collect technical information about your organization's use of the Pivotal Container Service (PKS) on a regular basis. Since PKS is jointly developed and sold by VMware and Pivotal, we will share this information with one another. Information collected under CEIP or Telemetry does not personally identify any individual.

Regardless of your selection in the **Usage Data** pane, a small amount of data is sent from Cloud Foundry Container Runtime (CFCR) to the PKS tile. However, that data is not shared externally.

To configure the **Usage Data** pane, perform the following steps:

1. Select the **Usage Data** side-tab.
2. Read the Usage Data description.

3. Make your selection.

    a. To join the program, select **Yes, I want to join the CEIP and Telemetry Program for PKS**.
    b. To decline joining the program, select **No, I do not want to join the CEIP and Telemetry Program for PKS**.

4. Click **Save**.

Note: If you join the CEIP and Telemetry Program for PKS, open your firewall to allow outgoing access to `https://vcsa.vmware.com/ph-prd` on port `443`.

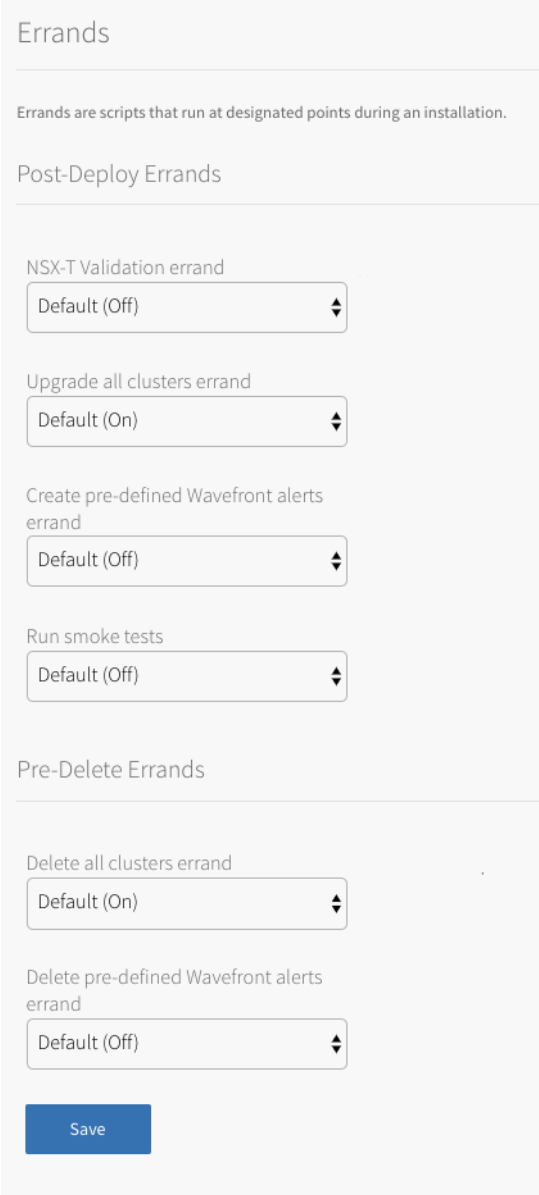
Errands

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

To configure when post-deploy and pre-delete errands for PKS are run, make a selection in the dropdown next to the errand.

We recommend that you set the **Run smoke tests** errand to **On**. The errand uses the PKS Command Line Interface (PKS CLI) to create a Kubernetes cluster and then delete it. If the creation or deletion fails, the errand fails and the installation of the PKS tile is aborted.

For the other errands, we recommend that you leave the default settings.



For more information about errands and their configuration state, see Managing Errands in Ops Manager.

WARNING: Because PKS uses floating stemcells, updating the PKS tile with a new stemcell triggers the rolling of every VM in each cluster. Also, updating other product tiles in your deployment with a new stemcell causes the PKS tile to roll VMs. This rolling is enabled by the **Upgrade all clusters errand**. We recommend that you keep this errand turned on because automatic rolling of VMs ensures that all deployed cluster VMs are patched. However, automatic rolling can cause downtime in your deployment.

If you are upgrading PKS, you must enable the **Upgrade All Clusters** errand.

Resource Config

To modify the resource usage of PKS and specify your PKS API load balancer, follow the steps below:

1. Select **Resource Config**.
2. In the **Load Balancers** column, enter the name of your PKS API load balancer. The name of your PKS API load balancer is `YOUR-ENVIRONMENT-NAME-pks-lb`. Refer to the environment name you configured in your `terraform.tfstate` file during Step 1: Download Templates and Edit Variables File. Then, append `-pks-lb` to that environment name.

   Note: After you click **Apply Changes** for the first time, BOSH assigns the PKS VM an IP address. BOSH uses the name you provide in the **Load Balancers** column to locate your load balancer, and then connect the load balancer to the PKS VM using its new IP address.

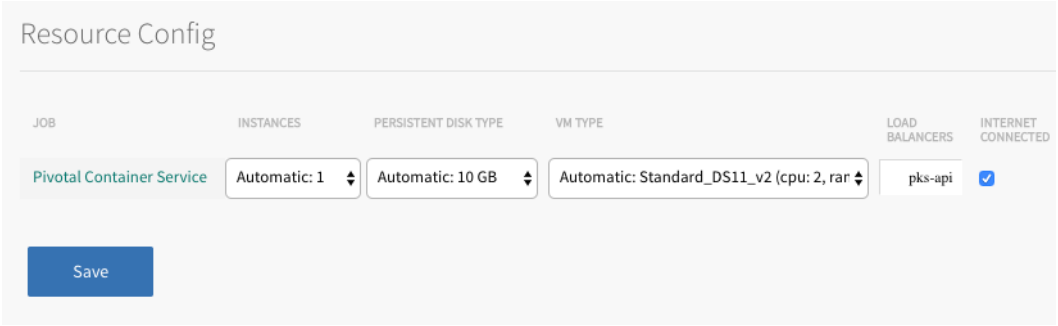
VMs used by **Pivotal Container Service** jobs must meet the following minimum requirements:

CPU	Memory	Disk
2	8 GB	29 GB

Note: If you experience timeouts or slowness when interacting with the PKS API, select a **VM Type** with greater CPU and memory resources.

To confirm you are deploying **Pivotal Container Service** job VMs meeting the minimum requirements, perform the following steps:

1. Select a **VM Type** with CPU, memory and disk resources either matching or exceeding the minimum **Pivotal Container Service** job VM requirements.



2. Select **Save**.

Step 3: Apply Changes

1. Return to the Ops Manager Installation Dashboard.
2. Click **Review Pending Changes**. Select the product that you intend to deploy and review the changes. For more information, see Reviewing Pending Product Changes.
3. Click **Apply Changes**.

Step 4: Retrieve the PKS API Endpoint

You must share the PKS API endpoint to allow your organization to use the API to create, update, and delete clusters. For more information, see Creating Clusters.

To retrieve the PKS API endpoint, do the following:

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the Pivotal Container Service tile.
3. Click the **Status** tab and locate the **Pivotal Container Service** job. The IP address of the Pivotal Container Service job is the PKS API endpoint.

Step 5: Configure an Azure Load Balancer for the PKS API

Follow the procedures in Configuring an Azure Load Balancer for the PKS API to configure an Azure load balancer for the PKS API.

Step 6: Install the PKS and Kubernetes CLIs

The PKS and Kubernetes CLIs help you interact with your PKS-provisioned Kubernetes clusters and Kubernetes workloads. To install the CLIs, follow the instructions below:

- Installing the PKS CLI
- Installing the Kubernetes CLI

Step 7: Configure PKS API Access

Follow the procedures in Configuring PKS API Access.

Step 8: Configure Authentication for PKS

Configure authentication for PKS using User Account and Authentication (UAA). For information, see Managing Users in PKS with UAA.

Next Steps

After installing PKS on Azure, you may want to do one or more of the following:

- Create a load balancer for your PKS clusters. For more information, see Creating and Configuring an Azure Load Balancer for PKS Clusters.
- Create your first PKS cluster. For more information, see Creating Clusters.

Please send any feedback you have to pks-feedback@pivotal.io.

Configuring an Azure Load Balancer for the PKS API

Page last updated:

This topic describes how to create a load balancer for the Pivotal Container Service (PKS) API using Azure.

Refer to the procedures in this topic to create a load balancer using Azure. To use a different load balancer, use this topic as a guide.

Prerequisites

To complete the steps below, you must identify the PKS API virtual machine (VM). You can find the name in the following ways:

- In the Azure Dashboard, locate the VM tagged with `instance_group:pivotal-container-service`.
- On the command line, run `bosh vms`.

Create Health Probe

1. From the Azure Dashboard, open the **Load Balancers** service.
2. In the **Settings** menu, select **Health probes**.
3. On the **Health probes** page, click **Add**.
4. On the **Add health probe** page, complete the form as follows:
    a. **Name**: Name the health probe.
    b. **Protocol**: Select **TCP**.
    c. **Port**: Enter `9021`.
    d. **Interval**: Enter the interval of time to wait between probe attempts.
    e. **Unhealthy Threshold**: Enter a number of consecutive probe failures that must occur before a VM is considered unhealthy.
5. Click **OK**.

Create Load Balancing Rule

1. From the Azure Dashboard, open the **Load Balancers** service.
2. In the **Settings** menu, select **Load Balancing Rules**.
3. On the **Load balancing rules** page, click **Add**.
4. On the **Add load balancing rules** page, complete the form as follows:
    a. **Name**: Name the load balancing rule.
    b. **IP Version**: Select **IPv4**.
    c. **Frontend IP address**: Select the appropriate IP address. Clients communicate with your load balancer on the selected IP address and service traffic is routed to the target VM by this NAT rule.
    d. **Protocol**: Select **TCP**.
    e. **Port**: Enter `9021`.
    f. **Backend port**: Enter `9021`.
    g. **Health Probe**: Select the health probe that you created in Create Health Probe.
    h. **Session persistence**: Select **None**.
5. Click **OK**.

Create Inbound Security Rule

1. From the Azure Dashboard, open the **Security Groups** service.

2. Click the name of the Security Group attached to the subnet where PKS API is deployed. If you deployed PKS using Terrform, the name of the Security Group ends with the suffix `bosh-deployed-vms-security-group`.
3. In the **Settings** menu for your security group, select **Inbound security rules**.
4. Click **Add**.
5. On the **Add inbound security rule** page, click **Advanced** and complete the form as follows:
    a. **Name**: Name the inbound security rule.
    b. **Source**: Select **Any**.
    c. **Source port range**: Enter `*`.
    d. **Destination**: Select **Any**.
    e. **Destination port range**: Enter `9021,8443`.
6. Click **OK**.

Verify Hostname Resolution

1. In a browser, log into Ops Manager.
2. Click the PKS tile.
3. Select **PKS API**.
4. Record the **API Hostname (FQDN)**.
5. Verify that the API hostname resolves to the IP address of the load balancer.

Next Step

After you have configured an Azure load balancer for the PKS API, complete the PKS installation by returning to the Install the PKS and Kubernetes CLIs step of *Installing PKS on Azure*.

Please send any feedback you have to pks-feedback@pivotal.io.

Installing the PKS CLI

Page last updated:

This topic describes how to install the Pivotal Container Service Command Line Interface (PKS CLI).

To install the PKS CLI, follow the procedures for your operating system to download the PKS CLI from Pivotal Network. Binaries are only provided for 64-bit architectures.

Mac OS X

1. Navigate to Pivotal Network and log in.
2. Click **Pivotal Container Service (PKS)**.
3. Select your desired release version from the **Releases** dropdown.
4. Click **PKS CLI**.
5. Click **PKS CLI - Mac** to download the Mac OS X binary.
6. Rename the downloaded binary file to `pks`.
7. On the command line, run the following command to make the PKS binary act as an executable file:

```
$ chmod +x pks
```

8. Move the binary file into your `PATH`.

Linux

1. Navigate to Pivotal Network and log in.
2. Click **Pivotal Container Service (PKS)**.
3. Select your desired release version from the **Releases** dropdown.
4. Click **PKS CLI**.
5. Click **PKS CLI - Linux** to download the Linux binary.
6. Rename the downloaded binary file to `pks`.
7. On the command line, run the following command to make the PKS binary executable:

```
$ chmod +x pks
```

8. Move the binary file into your `PATH`.

Windows

1. Navigate to Pivotal Network and log in.
2. Click **Pivotal Container Service (PKS)**.
3. Select your desired release version from the **Releases** dropdown.
4. Click **PKS CLI**.
5. Click **PKS CLI - Windows** to download the Windows executable file.

6. Rename the downloaded binary file to `pks.exe`.

7. Move the binary file into your `PATH`.

Please send any feedback you have to pks-feedback@pivotal.io.

Installing the Kubernetes CLI

Page last updated:

This topic describes how to install the Kubernetes Command Line Interface (kubectl).

To install kubectl, follow the procedures for your operating system to download kubectl from Pivotal Network. Binaries are only provided for 64-bit architectures.

Mac OS X

1. Navigate to Pivotal Network and log in.
2. Click **Pivotal Container Service (PKS)**.
3. Click **Kubectl CLIs**.
4. Click **kubectl CLI - Mac** to download the kubectl binary.
5. Rename the downloaded binary to `kubectl`.
6. On the command line, run the following command to make the kubectl binary executable:

```
$ chmod +x kubectl
```

7. Move the binary into your `PATH`. For example:

```
$ mv kubectl /usr/local/bin/kubectl
```

Linux

1. Navigate to Pivotal Network and log in.
2. Click **Pivotal Container Service (PKS)**.
3. Click **Kubectl CLIs**.
4. Click **kubectl CLI - Linux** to download the kubectl binary.
5. Rename the downloaded binary to `kubectl`.
6. On the command line, run the following command to make the kubectl binary executable:

```
$ chmod +x kubectl
```

7. Move the binary into your `PATH`. For example:

```
$ mv kubectl /usr/local/bin/kubectl
```

Windows

1. Navigate to Pivotal Network and log in.
2. Click **Pivotal Container Service (PKS)**.
3. Click **Kubectl CLIs**.
4. Click **kubectl CLI - Windows** to download the kubectl executable file.

5. Rename the downloaded binary to `kubectl.exe`.

6. Move the binary into your `PATH`.

Please send any feedback you have to pks-feedback@pivotal.io.

Upgrading PKS Overview

Page last updated:

This section describes how to upgrade the Pivotal Container Service (PKS) tile. See the following topics:

- What Happens During PKS Upgrades
- Upgrade Preparation Checklist for PKS v1.3
- Upgrading PKS
- Upgrading PKS with NSX-T
- Upgrading to PKS v1.3.6 and NSX-T 2.4.x
- Maintaining Workload Uptime
- Configuring the Upgrade Pipeline

Please send any feedback you have to pks-feedback@pivotal.io.

What Happens During PKS Upgrades

This topic explains what happens to Kubernetes clusters provisioned by Pivotal Container Service (PKS) during PKS upgrades.

Introduction

PKS enables you to upgrade either the PKS tile and all PKS-provisioned Kubernetes clusters or only the PKS tile.

- Upgrades of the PKS Tile and PKS-Provisioned Clusters
- Upgrades of the PKS Tile Only

During an upgrade of the PKS tile, your configuration settings are automatically migrated to the new tile version. For upgrading instructions, see Upgrading PKS.

Note: Upgrading from PKS v1.2.5+ to PKS v1.3.x causes all certificates to be automatically regenerated. The old certificate authority is still trusted, and has a validity of one year. But the new certificates are signed with a new certificate authority, which is valid for four years.

Canary Instances

The PKS tile is a BOSH deployment. When you deploy or upgrade a product using BOSH, the number of canary instances can affect the deployment.

BOSH-deployed products can set a number of canary instances to upgrade first, before the rest of the deployment VMs. BOSH continues the upgrade only if the canary instance upgrade succeeds. If the canary instance encounters an error, the upgrade stops running and other VMs are not affected.

The PKS tile uses one canary instance when deploying or upgrading PKS.

Upgrades of the PKS Tile and PKS-Provisioned Clusters

During an upgrade of the PKS tile and PKS-provisioned clusters, the following occurs:

1. The PKS API server is recreated. For more information, see PKS API Server.
2. Each of your Kubernetes clusters is recreated, one at a time. This includes the following stages for each cluster:
    a. Master nodes are recreated. For more information, see Master Nodes.
    b. Worker nodes are recreated. For more information, see Worker Nodes.

Note: When PKS is set to upgrade both the PKS tile and PKS-provisioned clusters, updating any stemcell in your deployment rolls every VM in each Kubernetes cluster. This ensures that all the VMs are patched. With the recommended resource configuration described above, no workload downtime is expected. For information about maintaining your Kubernetes workload uptime, see Maintaining Workload Uptime.

PKS API Server

When the PKS API server is recreated, you cannot interact with the PKS control plane or manage Kubernetes clusters. These restrictions prevent you from performing the following actions:

- Logging in through the PKS CLI
- Retrieving information about clusters
- Creating and deleting clusters
- Resizing clusters

Recreating the PKS API server does not affect deployed Kubernetes clusters and their workloads. You can still interact with them through the Kubernetes Command Line Interface, `kubectl`.

For more information about the PKS control plane, see PKS Control Plane Overview in *PKS Cluster Management*.

Master Nodes

When PKS recreates a single-master cluster during an upgrade, you cannot interact with your cluster, use `kubectl`, or push new workloads.

Note: To avoid this loss of functionality, Pivotal recommends using multi-master clusters.

Worker Nodes

When PKS recreates worker nodes, the upgrade runs on a single VM at a time. During the upgrade, the VM stops running containers. If your workloads run on a single VM, your apps will experience downtime.

When worker nodes are recreated, PKS upgrades Kubernetes to the version shipped with the PKS tile. See Enterprise PKS Release Notes.

Note: To avoid downtime for stateless workloads, Pivotal recommends using at least one worker node per availability zone (AZ). For stateful workloads, Pivotal recommends using a minimum of two worker nodes per AZ.

Upgrades of the PKS Tile Only

During an upgrade of the PKS tile only, the PKS API server is recreated.

When the PKS API server is recreated, you cannot interact with the PKS control plane or manage Kubernetes clusters. These restrictions prevent you from performing the following actions:

- Logging in through the PKS CLI
- Retrieving information about clusters
- Creating and deleting clusters
- Resizing clusters

Recreating the PKS API server does not affect deployed Kubernetes clusters and their workloads. You can still interact with them through the Kubernetes Command Line Interface, `kubectl`.

To upgrade the PKS tile only, set the **Upgrade all clusters errand** to **Off** before you begin the upgrade. For more information, see Upgrade the PKS Tile in *Upgrading PKS*.

For more information about the PKS control plane, see PKS Control Plane Overview in *PKS Cluster Management*.

Note: When PKS is set to upgrade only the PKS tile and not the clusters, the Kubernetes cluster version falls behind the PKS tile version. If the clusters fall more than one version behind the tile, PKS cannot upgrade the clusters. The clusters must be upgraded to match the PKS tile version before the next tile upgrade.

Please send any feedback you have to pks-feedback@pivotal.io.

Upgrade Preparation Checklist for PKS v1.3

This topic serves as a checklist for preparing to upgrade Pivotal Container Service (PKS) from v1.2 to v1.3.

This topic contains important preparation steps that you must follow before beginning your upgrade. Failure to follow these instructions may jeopardize your existing deployment data and cause the upgrade to fail.

After completing the steps in this topic, you can continue to Upgrading PKS. If you are upgrading PKS for environments using vSphere with NSX-T, continue to Upgrading PKS with NSX-T.

Back Up Your PKS Deployment

We recommend backing up your PKS deployment before upgrading, to restore in case of failure.

If you are upgrading PKS for environments using vSphere with NSX-T, back up your environment using the procedures in the following topics:

- Backup PKS
- Backup NSX-T
- Backup vCenter

> **Note**: If you choose not to back up PKS, NSX-T, or vCenter, we recommend backing up the NSX-T and NSX-T Container Plugin (NCP) logs.

If you are upgrading PKS for any other IaaS, back up the PKS v1.2 control plane. For more information, see Backing Up and Restoring PKS.

Review Changes in PKS v1.3

Review the Release Notes for the version or versions of PKS you are upgrading to.

Understand What Happens During PKS Upgrades

Review What Happens During PKS Upgrades, and consider your workload capacity and uptime requirements.

View Workload Resource Usage

View your workload resource usage in Dashboard. For more information, see Accessing Dashboard.

If workers are operating too close to their capacity, the PKS upgrade can fail. To prevent workload downtime during a cluster upgrade, Pivotal recommends running your workload on at least three worker VMs, using multiple replicas of your workloads spread across those VMs. For more information, see Maintaining Workload Uptime.

If your clusters are near capacity for your existing infrastructure, we recommend scaling up your clusters before you upgrade. Scale up your cluster by running `pks resize` or create a cluster using a larger plan. For more information, see Scaling Existing Clusters.

Verify Health of Kubernetes Environment

Verify that your Kubernetes environment is healthy. To verify the health of your Kubernetes environment, see Verifying Deployment Health.

Verify NSX-T Configuration (vSphere with NSX-T Only)

If you are upgrading PKS for environments using vSphere with NSX-T, perform the following steps:

1. Make sure there are no issues with vSphere by following the steps below:

a. Verify that datastores have enough space.
b. Verify that hosts have enough memory.
c. Verify that there are no alarms.
d. Verify that hosts are in a good state.

2. Verify that NSX Edge is configured for high availability using Active/Standby mode.

> **Note**: Workloads in your Kubernetes cluster are unavailable while the NSX Edge nodes run the upgrade unless you configure NSX Edge for high availability. For more information, see the Configure NSX Edge for High Availability (HA) section of *Preparing NSX-T Before Deploying PKS*.

Clean Up Failed Kubernetes Clusters

Clean up previous failed attempts to delete PKS clusters with the PKS Command Line Interface (PKS CLI).

Perform the following steps:

1. View your deployed clusters by running the following command:

```
pks clusters
```

If the `Status` of any cluster displays as `FAILED`, continue to the next step. If no cluster displays as `FAILED`, no action is required. Continue to the next section.

2. Perform the procedures in Cannot Re-Create a Cluster that Failed to Deploy to clean up the failed BOSH deployment.

3. Run `pks clusters` to view your deployed clusters again. If any clusters remain in the `FAILED` state, contact PKS Support.

Verify Kubernetes Clusters Have Unique External Hostnames

Verify that existing Kubernetes clusters have unique external hostnames by checking for multiple Kubernetes clusters with the same external hostname.

Perform the following steps:

1. Log in to the PKS CLI. For more information, see Logging in to PKS. You must log in with an account that has the UAA scope of `pks.clusters.admin`. For more information about UAA scopes, see Managing Users in Enterprise PKS with UAA.

2. View your deployed PKS clusters by running the following command:

```
pks clusters
```

3. For each deployed cluster, run `pks cluster CLUSTER-NAME` to view the details of the cluster. For example:

```
$ pks cluster my-cluster
```

Examine the output to verify that the `Kubernetes Master Host` is unique for each cluster.

Verify PKS Proxy Configuration

Verify your current PKS proxy configuration.

Perform the following steps:

1. Check whether an existing proxy is enabled:

a. Log in to Ops Manager.
b. Click the **Pivotal Container Service** tile.
c. Click **Networking**.
d. If **HTTP/HTTPS Proxy** is **Disabled**, no action is required. Continue to the next section.

If **HTTP/HTTPS Proxy** is **Enabled**, continue to the next step.

2. If the existing **No Proxy** field contains any of the following values, or you plan to add any of the following values, contact PKS Support:
    - `localhost`
    - Hostnames containing dashes, such as `my-host.mydomain.com`

Check PodDisruptionBudget Value

A PKS upgrade will run endlessly and never complete if any Kubernetes application, for any reason, has had a `PodDisruptionBudget` created for it with `maxUnavailable` set to `0`.

Perform the following steps:

1. Use the Kubernetes CLI, `kubectl`, to verify the `PodDisruptionBudget` as the cluster administrator. Run the following command:

    ```
    kubectl get poddisruptionbudgets --all-namespaces
    ```

    Note: For more information about kubectl, see Overview of kubectl in the Kubernetes documentation.

2. Examine the output. Any app listed should not show `0` in the `MAX UNAVAILABLE` column.

Upgrading PKS

Page last updated:

This topic explains how to upgrade the Pivotal Container Service (PKS) tile and existing Kubernetes clusters.

The supported upgrade paths to PKS v1.3.x are from PKS v1.2.5 and later. PKS v1.3.x is compatible with Ops Manager v2.3.1 or later and Ops Manager v2.4.x.

For conceptual information about upgrading the PKS tile and PKS-provisioned Kubernetes clusters, see What Happens During PKS Upgrades.

For information about upgrading PKS on vSphere with NSX-T integration, see Upgrading PKS with NSX-T.

WARNING: Do not manually upgrade your Kubernetes version. The PKS service includes the compatible Kubernetes version.

Note: Upgrading from PKS v1.2.5+ to PKS v1.3.x causes all certificates to be automatically regenerated. The old certificate authority is still trusted, and has a validity of one year. But the new certificates are signed with a new certificate authority, which is valid for four years.

Before You Upgrade

This section describes the activities you must perform before upgrading PKS.

Determine Your Upgrade Path

Use the following table to determine your upgrade path to PKS v1.3.x.

If your current version of PKS is…	Then use the following upgrade path:
v1.1.4 or earlier	1. Upgrade to PKS v1.1.5 or later. 2. Upgrade to Ops Manager v2.3.10 or later. 3. Upgrade to PKS v1.2.5. 4. (Optional) Upgrade to Ops Manager v2.4.4 or later. 5. Upgrade to PKS v1.3.x.
v1.1.5 to v1.2.4	1. Upgrade to Ops Manager v2.3.10 or later. 2. Upgrade to PKS v1.2.5. 3. (Optional) Upgrade to Ops Manager v2.4.4 or later. 4. Upgrade to PKS v1.3.x.
v1.2.5 or later	1. Upgrade to Ops Manager v2.3.10 or later. 2. (Optional) Upgrade to Ops Manager v2.4.4 or later. 3. Upgrade to PKS v1.3.x.

Note: Upgrading an existing Ops Manager installation from v2.3.1 or later to v2.3.10, or v2.4.2 or later to v2.4.4, is required only for supporting Azure PKS v1.3.x deployments.

Prepare to Upgrade

If you have not already, complete the steps in the Upgrade Preparation Checklist for PKS v1.3.

During the Upgrade

This section describes the steps required to upgrade to PKS v1.3.x.

Step 1: Upgrade to PKS v1.2.5 or Later

Skip this step if you are already running PKS v1.2.5+.

Follow the procedures detailed in Upgrading PKS in the PKS v1.2 documentation.

Step 2: Upgrade to Ops Manager v2.3.1+ or v2.4.x

Before you upgrade to PKS v1.3.x, you must upgrade to Ops Manager v2.3.1+ or v2.4.x.

1. Follow the procedures detailed in Upgrade Ops Manager and Installed Products to v2.3 or Upgrade Ops Manager and Installed Products to v2.4 .
2. Verify that the PKS control plane remains functional by performing the following steps:
    a. Add more workloads and create an additional cluster. For more information about performing those actions, see About Workload Upgrades in *Maintaining Workload Uptime* and Creating Clusters.
    b. Monitor the PKS control plane VM by clicking the **Pivotal Container Service tile**, selecting **Status** tab, and reviewing the **Pivotal Container Service** VM's data points. If any data points are at capacity, scale your deployment accordingly.

Step 3: Upgrade to PKS v1.3.x

To upgrade to PKS v1.3.x, follow the same Ops Manager process that you use to install the tile for the first time.

Your configuration settings migrate to the new version automatically. Follow the steps below to perform an upgrade.

1. Review the Release Notes for the version you are upgrading to.
2. Download the desired version of the product from Pivotal Network.
3. Navigate to the Ops Manager Installation Dashboard and click **Import a Product** to upload the product file.
4. Under the **Import a Product** button, click **+** next to **Pivotal Container Service**. This adds the tile to your staging area.

Step 4: Download and Import the Stemcell

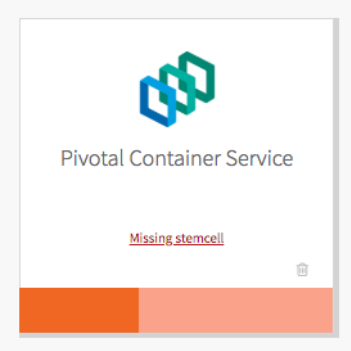
PKS v1.3.x uses a Xenial stemcell.

If Ops Manager does not have the Xenial stemcell required for PKS, the PKS tile displays the message **Missing stemcell**.

> **Note:** If the **Stemcell Library** in Ops Manager already has a compatible Xenial stemcell, the **Missing stemcell** link does not appear. You do not need to download or import a new stemcell and can skip this step.

To download and import a new Xenial stemcell, follow the steps below:

1. On the **Pivotal Container Service** tile, click on the **Missing stemcell** link.



2. In the **Stemcell Library**, locate Pivotal Container Service and note the required stemcell version.
3. Visit the Stemcells for PCF (Ubuntu Xenial) page on Pivotal Network, and download the required stemcell version appropriate for your IaaS.
4. Return to the **Installation Dashboard** in Ops Manager, and click on **Stemcell Library**.
5. On the **Stemcell Library** page, click **Import Stemcell** and select the stemcell file you downloaded from Pivotal Network.
6. Select Pivotal Container Service and click **Apply Stemcell to Products**.
7. Verify that Ops Manager successfully applied the stemcell. The stemcell version you imported and applied appears in the **Staged** column for Pivotal Container Service.
8. Select the **Installation Dashboard** link to return to the Installation Dashboard.

Step 5: Verify Errand Configuration

To verify that errands are configured correctly in the PKS tile, perform the following steps.

1. Click the newly-added **Pivotal Container Service** tile.
2. Click **Errands**.
3. Under **Post-Deploy Errands**, verify that the **Upgrade all clusters errand** is set to **Default (On)**. The errand upgrades a single Kubernetes cluster at a time. Upgrading PKS Kubernetes clusters can temporarily interrupt the service, as described in Service Interruptions.

   WARNING: If you are upgrading PKS, you must enable the **Upgrade All Clusters** errand.

4. Under **Post-Deploy Errands**, set the **Run smoke tests** errand to **On**. The errand uses the PKS Command Line Interface (PKS CLI) to create a Kubernetes cluster and then delete it. If the creation or deletion fails, the errand fails and the installation of the PKS tile is aborted.
5. Review the other configuration panes. Click **Save** on any panes where you make changes.

   Note: When you upgrade PKS, you must place singleton jobs in the AZ you selected when you first installed the PKS tile. You cannot move singleton jobs to another AZ.

Step 6: Apply Changes to the PKS Tile

Perform the following steps to complete the upgrade to the PKS tile.

1. Return to the **Installation Dashboard** in Ops Manager.
2. Click **Review Pending Changes**. For more information about this Ops Manager page, see Reviewing Pending Product Changes.
3. Click **Apply Changes**.
4. (Optional) To monitor the progress of the **Upgrade all clusters errand** using the BOSH CLI, do the following:
   a. Log in to the BOSH Director by running `bosh -e MY-ENVIRONMENT log-in` from a VM that can access your PKS deployment. For more information, see Managing PKS Deployments with BOSH.

b. Run `bosh -e MY-ENVIRONMENT tasks`.
c. Locate the task number for the errand in the # column of the BOSH output.
d. Run `bosh task TASK-NUMBER`, replacing `TASK-NUMBER` with the task number you located in the previous step.

After the Upgrade

After you complete the upgrade to PKS v1.3.x, complete the following verifications and upgrades.

Update PKS and Kubernetes CLIs

Update the PKS and Kubernetes CLIs on any local machine where you run commands that interact with your upgraded version of PKS.

To update your CLIs, download and re-install the PKS and Kubernetes CLI distributions that are provided with PKS on Pivotal Network.

For more information about installing the CLIs, see the following topics:

- Installing the PKS CLI
- Installing the Kubernetes CLI

Verify the Upgrade

After you apply changes to the PKS tile and the upgrade is complete, perform the following steps:

1. Verify that your Kubernetes environment is healthy. To verify the health of your Kubernetes environment, see Verifying Deployment Health.

2. Verify that the PKS control plane remains functional by performing the following steps:

    a. Add more workloads and create an additional cluster. For more information about performing those actions, see About Workload Upgrades in *Maintaining Workload Uptime* and Creating Clusters.
    b. Monitor the PKS control plane VM by clicking the **Pivotal Container Service tile**, selecting **Status** tab, and reviewing the **Pivotal Container Service** VM's data points. If any data points are at capacity, scale your deployment accordingly.

(Optional) Upgrade vSphere

If you are deploying PKS on vSphere, consult the chart below, and upgrade vSphere if necessary.

Versions	Editions
• VMware vSphere 6.7 U1 EP06 (ESXi670-201901001) – for NSX-T 2.4 • VMware vSphere 6.7 U1 • VMware vSphere 6.7.0 • VMware vSphere 6.5 U2 P03 (ESXi650-201811002) – for NSX-T 2.4 • VMware vSphere 6.5 U2 • VMware vSphere 6.5 U1	• vSphere Enterprise Plus • vSphere with Operations Management Enterprise Plus

Note: VMware vSphere 6.7 is only supported with Ops Manager v2.3.1 or later.

Please send any feedback you have to pks-feedback@pivotal.io.

Upgrading PKS with NSX-T

Page last updated:

This topic explains how to upgrade the Pivotal Container Service (PKS) for environments using vSphere with NSX-T.

The supported upgrade paths to PKS v1.3.x are from PKS v1.2.5 and later. PKS v1.3.x is compatible with Ops Manager v2.3.1 or later and Ops Manager v2.4.x.

Note: Upgrading from PKS v1.2.5+ to PKS v1.3.x causes all certificates to be automatically regenerated. The old certificate authority is still trusted, and has a validity of one year. But the new certificates are signed with a new certificate authority, which is valid for four years.

Before You Upgrade

This section describes the activities you must perform before upgrading PKS.

Consult Compatibility Charts

For information about PKS with NSX-T and Ops Manager compatibility, refer to the release notes.

Determine Your Upgrade Path

For information about the supported upgrade path, refer to the release notes.

Prepare to Upgrade

If you have not already, complete the steps in the Upgrade Preparation Checklist for PKS v1.3.

During the Upgrade

This section describes the steps required to upgrade to PKS v1.3 with NSX-T v2.3.

Step 1: Upgrade to PKS v1.1.5 or Later

Skip this step if you are already running PKS v1.1.5+.

Follow the procedures detailed in Upgrading PKS with NSX-T in the PKS v1.1 documentation.

Note: PKS v1.1.5 with NSX-T introduces architectural changes that require larger sized worker node VMs. Before you upgrade to PKS v1.1.5 or later, you must increase the size of the Kubernetes worker node VM. For more information on how to increase the worker node VM size, see Increase the Kubernetes Worker Node VM Size in the PKS v1.1 documentation. For more information about the architectural changes in PKS v1.1.5 with NSX-T, see NSX-T Architectural Changes in the PKS v1.1.5 Release Notes.

Step 2: Upgrade to NSX-T v2.2

Skip this step if you are already running NSX-T v2.2.

To upgrade to NSX-T v2.2, follow the procedures detailed in Upgrading NSX-T in the VMware documentation.

Step 3: Upgrade to v2.3.1+

Before you upgrade to PKS v1.2.5, you must upgrade to Ops Manager v2.3.1+.

1. Follow the procedures detailed in Upgrade Ops Manager and Installed Products to v2.3 or Upgrade Ops Manager and Installed Products to v2.4 .

2. Verify that the PKS control plane remains functional by performing the following steps:

    a. Add more workloads and create an additional cluster. For more information about performing those actions, see About Workload Upgrades in *Maintaining Workload Uptime* and Creating Clusters.
    b. Monitor the PKS control plane VM by clicking the **Pivotal Container Service tile**, selecting **Status** tab, and reviewing the **Pivotal Container Service** VM's data points. If any data points are at capacity, scale your deployment accordingly.

Step 4: Upgrade to PKS v1.3.x

To upgrade PKS, you follow the same Ops Manager process that you use to install the tile for the first time.

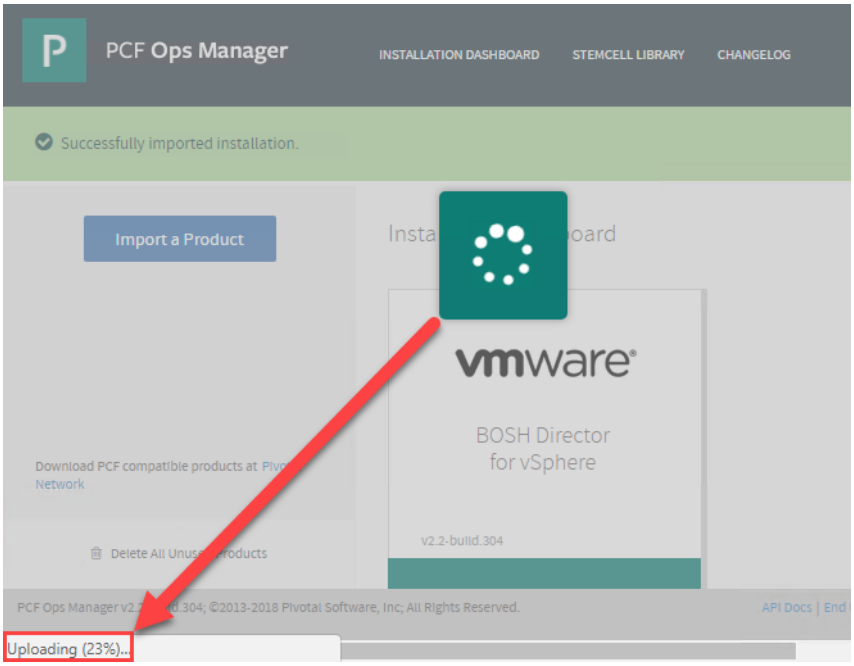
Your configuration settings migrate to the new version automatically. Follow the steps below to perform an upgrade.

1. Review the Release Notes for the version you are upgrading to.

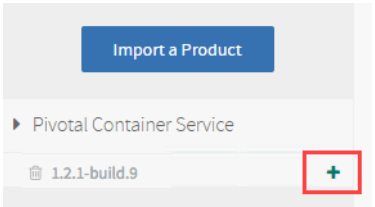
2. Download the desired version of the product from Pivotal Network.

3. Navigate to the Ops Manager Installation Dashboard and click **Import a Product**.

4. Browse to the PKS product file and select it. Uploading the file takes several minutes.



5. Under the **Import a Product** button, click + next to **Pivotal Container Service**. This adds the tile to your staging area.



Step 5: Download and Import the Stemcell

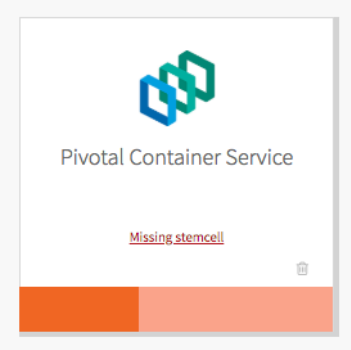
PKS v1.3.x uses a Xenial stemcell.

If Ops Manager does not have the Xenial stemcell required for PKS, the PKS tile displays the message **Missing stemcell**.

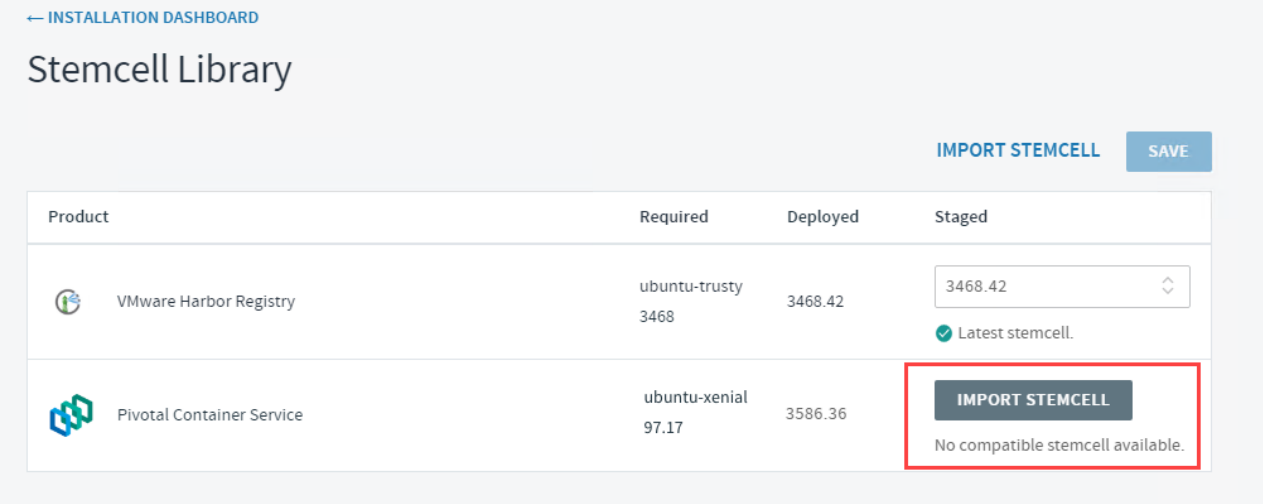
> **Note:** If the **Stemcell Library** in Ops Manager already has a compatible Xenial stemcell, the **Missing stemcell** link does not appear. You do not need to download or import a new stemcell and can skip this step.

To download and import a new Xenial stemcell, follow the steps below:

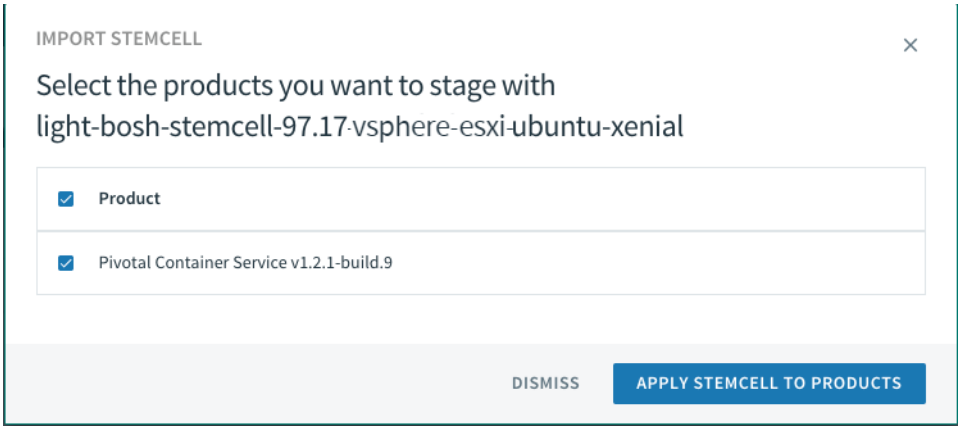
1. On the **Pivotal Container Service** tile, click on the **Missing stemcell** link.



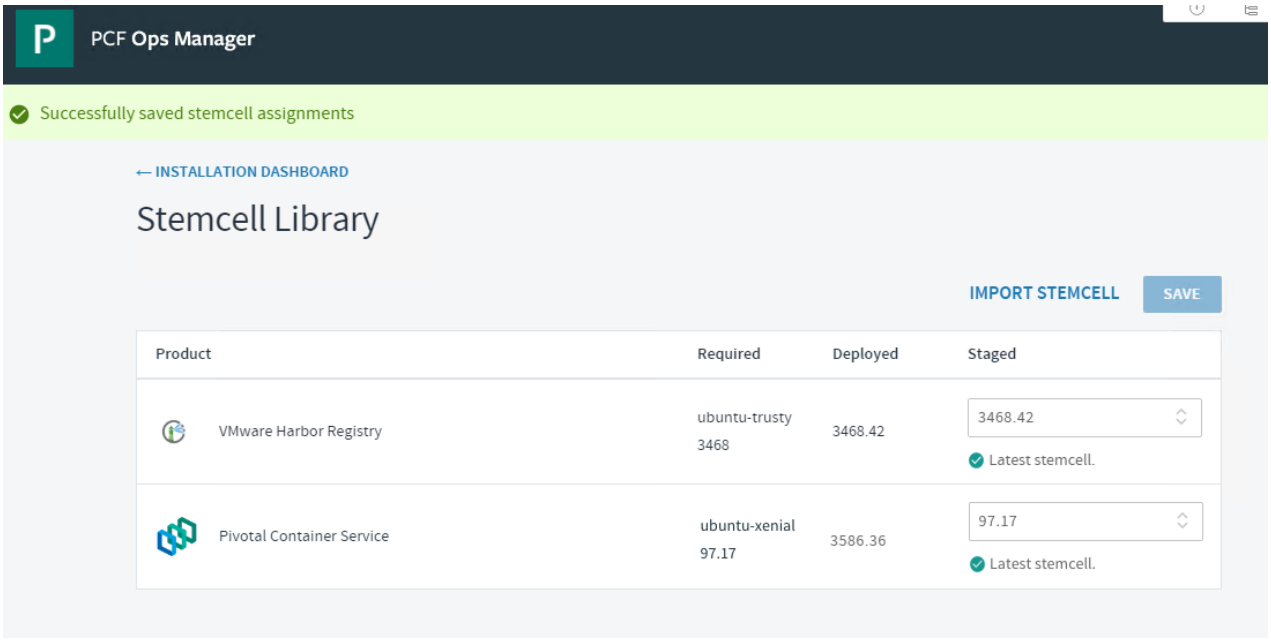
2. In the **Stemcell Library**, locate Pivotal Container Service and note the required stemcell version.
3. Visit the Stemcells for PCF (Ubuntu Xenial) page on Pivotal Network, and download the required stemcell version for vSphere.
4. Return to the **Installation Dashboard** in Ops Manager, and click on **Stemcell Library**.
5. On the **Stemcell Library** page, click **Import Stemcell** and select the stemcell file you downloaded from Pivotal Network.



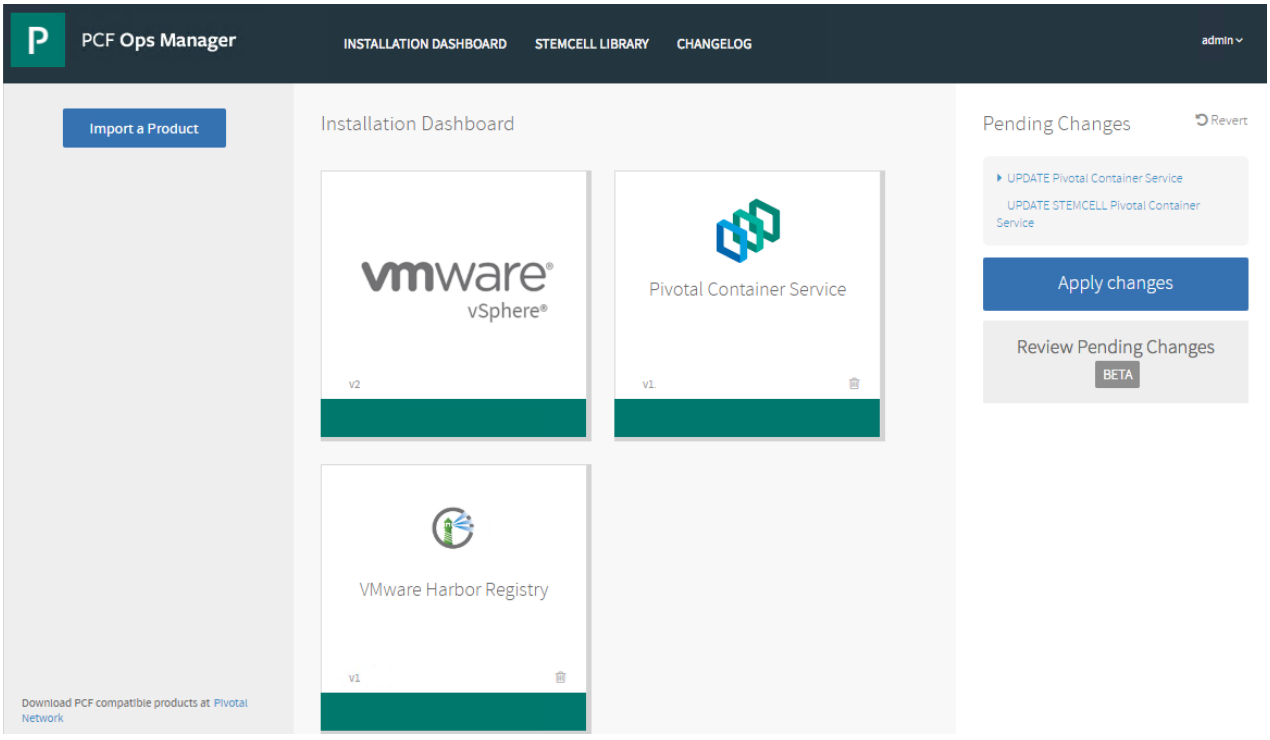
6. Select the PKS product and click **Apply Stemcell to Products**.



7. Verify that Ops Manager successfully applied the stemcell.



8. Select the **Installation Dashboard** link to return to the Installation Dashboard.



Step 6: Verify Errand Configuration

To verify that errands are configured correctly in the PKS tile, perform the following steps.

1. In the PKS tile, click **Errands**.

2. Under **Post-Deploy Errands**, verify that the listed errands are configured as follows:

    - **NSX-T Validation errand**: Set to **On**
    - **Upgrade all clusters errand**: Set to **Default (On)**
    - **Create pre-defined Wavefront alerts errand**: Set to **Default (Off)**
    - **Run smoke tests**: Set to **On**. The errand uses the PKS Command Line Interface (PKS CLI) to create a Kubernetes cluster and then delete it. If the creation or deletion fails, the errand fails and the installation of the PKS tile is aborted.

> **WARNING**: If you set the **Upgrade all clusters errand** to **Off**, your Kubernetes cluster version will fall behind the PKS tile version. If your clusters fall more than one version behind the tile, you can no longer upgrade the clusters. You must upgrade your clusters to match the PKS tile version before the next tile upgrade.

3. If you make any changes, click **Save**.

Step 6: Apply Changes to the PKS Tile

Perform the following steps to complete the upgrade to the PKS tile.

1. Return to the **Installation Dashboard** in Ops Manager.
2. Click **Review Pending Changes**. For more information about this Ops Manager page, see Reviewing Pending Product Changes.
3. Click **Apply Changes**.

Step 7: Upgrade to NSX-T v2.3

NSX-T v2.3 is the recommended version of NSX-T to use with PKS v1.3.

To upgrade to NSX-T v2.3, follow the procedures detailed in Upgrading NSX-T Data Center.

After the Upgrade

After you complete the upgrade to PKS v1.3.x and NSX-T v2.3, complete the following verifications and upgrades.

Update PKS and Kubernetes CLIs

Update the PKS and Kubernetes CLIs on any local machine where you run commands that interact with your upgraded version of PKS.

To update your CLIs, download and re-install the PKS and Kubernetes CLI distributions that are provided with PKS on Pivotal Network.

For more information about installing the CLIs, see the following topics:

- Installing the PKS CLI
- Installing the Kubernetes CLI

Verify the Upgrade

After you apply changes to the PKS tile and the upgrade is complete, verify that your Kubernetes environment is healthy and confirm that NCP is running on the master node VM.

To verify the health of your Kubernetes environment and NCP, see Verifying Deployment Health.

(Optional) Upgrade vSphere

If you are deploying PKS on vSphere with NSX–T, consult the chart below, and upgrade vSphere if necessary. Upgrade vSphere from version 6.5 or 6.5 U1 to 6.5 U2 or 6.7.

Versions	Editions
• VMware vSphere 6.7 U1 EP06 (ESXi670-201901001) – for NSX-T 2.4 • VMware vSphere 6.7 U1 • VMware vSphere 6.7.0 • VMware vSphere 6.5 U2 P03 (ESXi650-201811002) – for NSX-T 2.4	• vSphere Enterprise Plus • vSphere with Operations Management Enterprise Plus

• VMware vSphere 6.5 U2 • VMware vSphere 6.5 U1	

Note: VMware vSphere 6.7 is only supported with Ops Manager v2.3.1 or later and NSX−T v2.3.

For more information, see Upgrading vSphere in an NSX Environment in the VMware documentation.

Please send any feedback you have to pks-feedback@pivotal.io.

Upgrading PKS with NSX-T to NSX-T v2.4.0.1

Page last updated:

This topic describes how to upgrade your PKS with NSX-T environment from NSX-T v2.3 to v2.4.

Step 0: Prepare to Upgrade

Review related documentation in preparation for the upgrade of PKS:

1. Review the PKS Release Notes for the supported upgrade path and known issues.
2. Review the VMware Product Interoperability Matrix for PKS in the VMware documentation.
3. Review the NSX-T 2.4 release notes.

Step 1: Upgrade to PKS v1.3.6

Upgrade the PKS tile from a supported version to to PKS v1.3.6. When you upgrade the PKS tile, the target version of NCP is installed (v2.4.0 in this case). This must be done before you upgrade to NSX-T v2.4.x.

If you are performing the upgrade during a maintenance window, it is not necessary to upgrade the Kubernetes clusters at this time, so you can deselect the upgrade all clusters errand for PKS. However, if you want your Kubernetes clusters to be upgraded immediately, ensure that the upgrade all clusters errand is enabled.

To upgrade the PKS tile to v1.3.6:

1. Download the PKS v1.3.6 tile from the Pivotal Network.
2. Upload the PKS v1.3.6 tile to Ops Manager.
3. Stage the 1.3.6 tile for deployment.
4. Review pending changes.
5. Apply changes.

Step 2: Verify Supported vSphere Versions and Required ESXi Patches

NSX-T v2.4.x supports the following vSphere versions with patches:

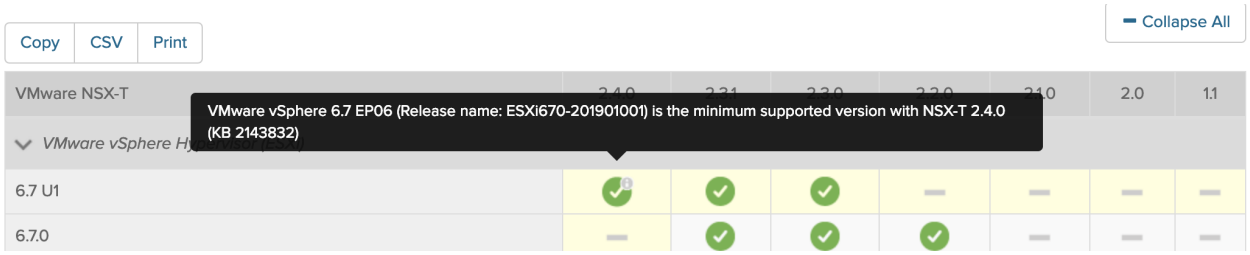
- VMware vSphere 6.7 EP06 (Release name: ESXi670-201901001) is the minimum supported version with NSX-T 2.4.0 (KB 2143832)
- VMware vSphere 6.5 P03 (Release Name: ESXi650-201811002) is the minimum supported version with NSX-T 2.4.0 (KB 2143832)

To verify the installation of supported vSphere versions and ESXi patches, perform the following steps:

1. Refer to the VMware Product Interoperability Matrices.

VMware NSX-T	2.4.0	2.3.1	2.3.0	2.2.0	2.1.0	2.0	1.1
VMware vSphere Hypervisor (ESXi)							
6.7 U1	✓	✓	✓	—	—	—	—
6.7.0	—	✓	✓	✓	—	—	—
6.5 U2	✓	✓	✓	✓	—	—	—
6.5 U1	—	✓	✓	✓	✓	✓	—
6.5.0	—	—	—	—	✓	✓	✓
6.0 U3	—	—	—	—	—	—	✓
6.0.0 U2	—	—	—	—	—	—	✓

2. Hover over the Information icon for vSphere 6.7 U1 and NSX-T 2.4: version-specific compatibility information is displayed. For example, see the message "*VMware vSphere 6.7 EP06 (Release name: ESXi670-201901001) is the minimum supported version with NSX-T 2.4.0 (KB 2143832)*" below:



   For details on the ESXi v6.7 U1 EP06 patch, refer to the VMware KB article Build numbers and versions of VMware ESXi/ESX.

3. Apply required ESXi patch upgrades:

   - To perform the ESXi patch upgrade using vCenter, refer to the vSphere Upgrade Manager documentation for guidance on applying the patch. See also the VMware ESXi Upgrade documentation for additional details.
   - To patch ESXi hosts in an air-gapped environment, use Zip files as described in the VMware ESXi documenation.

Step 3: Upgrade from NSX-T v2.3.1 to NSX-T v2.4

Upgrade NSX-T from v2.3.1 to v2.4.0.1.

1. To upgrade NSX-T from v2.3.1 to v2.4.0.1, refer to Upgrading NSX-T Data Center in the VMware documentation.

Note: You must use at least version v2.4.0.1 due to the following known issue in v2.4.0: Important information before upgrading to NSX-T Data Center 2.4.0 (67449). See the **Upgrade Path** section of the Release Notes for information on obtaining the hot-patch.

Note: When upgrading NSX-T, at the stage that the ESXi Transport Nodes are upgraded ("Hosts"), you may want to create a different host group for each ESXi host in the correct order so that hosts in maintenance mode only get upgraded. In vCenter, put each EXSi Transport Node (TN) host into maintenance mode, 1 at a time. Create the host group for that ESXi host and upgrade only it, then remove it from maintenance mode. Repeat this process for all ESXi TN hosts.

Note: Once you upgrade to NSX-T 2.4, the T0 router(s) and all other management plane objects can be seen only from the *Advanced Networking Configuration* tab. They will not be migrated to the new Policy UI.

Note: There are architectural changes in NSX 2.4. The NSX Controller is now a component of the NSX Manager. Once the NSX-T upgrade is complete, you will have a single NSX-T Manager node. Power off the NSX Controllers. At the end of the upgrade, you can delete the NSX Controller VMs. For more information, see Delete NSX Controllers in the NSX-T documentation.

Note: Once the upgrade to NSX 2.4 is complete, you may want to verify that your PKS environment is functioning properly by logging in to PKS and creating a small test cluster. If you cannot do this, troubleshoot the upgrade before proceeding. For more information, see Troubleshooting Upgrade Failures in the NSX-T documentation.

Step 4: Deploy Two Additional NSX Managers

With NSX-T v2.4, the NSX Controller component is now part of the NSX Manager. Previously the NSX Manager was a singleton, and HA was achieved using multiple NSX Controllers. With NSX-T v2.4, since the standalone NSX Controller component is no longer used, to achieve HA you need to deploy multiple (three) NSX Managers.

1. To deploy additional NSX Managers, refer to the Upgrading NSX-T Data Center documentation for guidance.

Note: When you add additional NSX Managers, the system prompts you to enter a Compute Manager, which is a vCenter Server. For more information, see Add a Compute Manager in the NSX-T documentation.

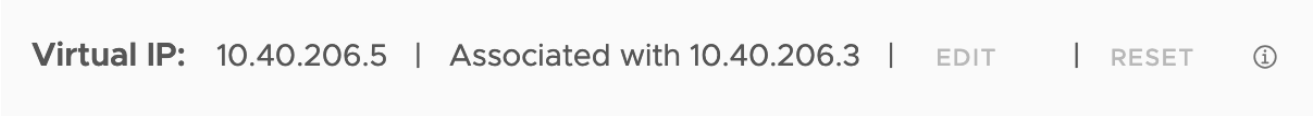
Step 5: Configure the NSX Manager VIP

Since you have deployed two additional NSX Managers (for a total of three), you need create a virtual IP address that can be used as a single endpoint to access the NSX Management cluster.

To create a VIP for the NSX Management cluster:

1. Log in to the NSX Manager interface.
2. Go to **System** > **Overview**.
3. Select **Virtual IP** > **Edit**.
4. Enter a publicly routable IP address, such as `10.40.206.5`.
5. Click **Save**.

   At this point in time, you can connect to any NSX-T manager using its own IP address, or use the VIP to connect to NSX-T Manager. Both methods work. However, note that the VIP is associated with a single NSX Manager.
6. To determine which NSX Manager the VIP is associated with, select the Virtual IP.



Step 6: Generate and Register a New NSX Manager CA Cert with the Cluster API

Both the BOSH Director tile and the PKS tile expect the NSX Manager CA certificate. However, the current NSX Manager CA certificate is associated with the original NSX Manager IP address. You need to generate a new NSX Manager CA cert using the VIP address, then register this certificate with NSX-T using the Cluster Certificate API:

1. To generate a new NSX Manage CA certificate and private key using the VIP address, follow the instructions in the Generate NSX CA Cert PKS documentation. Make sure you use the VIP address, such as `10.40.206.5` in our example above.
2. Import the new CA certificate to the NSX Manager. Refer to Import the Certificate to NSX Manager for instructions on doing this.
3. Register this certificate with the NSX Management cluster using a cURL command against the Cluster Certificate API.

   Note: In general the instructions provided in the Register the Certificate with NSX Manager documentation can be followed, with the exception that API endpoint is changed to the Cluster Certificate API.
4. Create environment variables for the VIP address and the certificate ID:

```
export NSX_MANAGER_IP_ADDRESS=10.40.206.5

export CERTIFICATE_ID="63bb6646-052c-49df-b603-64d7e5bdb5bf"
```

5. To register the new NSX-T Manager CA cert with the NSX Manager, run the following Cluster Certificate API command:

```
curl --insecure -u admin:'PASSWORD' -X POST "https://$NSX_MANAGER_IP_ADDRESS/api/v1/cluster/api-certificate?action=set_cluster_certificate&certificate_id=$CERTIFICATE_ID"
```

Where `PASSWORD` is your NSX Manager admin account password.

For example:

```
export NSX_MANAGER_IP_ADDRESS=198.51.100.4
export CERTIFICATE_ID="73bb66t6-0523-51df-k603-64g9g5bdb5rl"
curl --insecure -u admin:'PASSWORD' -X POST "https://$NSX_MANAGER_IP_ADDRESS/api/v1/cluster/api-certificate? \
action=set_cluster_certificate&certificate_id=$CERTIFICATE_ID"
```

6. To register the new certificate with your other NSX-T Manager appliances, repeat the process for each appliance, using each NSX Manager's IP address as $NSX_MANAGER_IP_ADDRESS.
7. To verify, using a browser go to the VIP address of the NSX Manager. Login and check that the new cert is used by the site (accessed using the VIP address).
8. To further verify, SSH to each NSX Manager host and run the following two commands. All certificates returned should be the same.

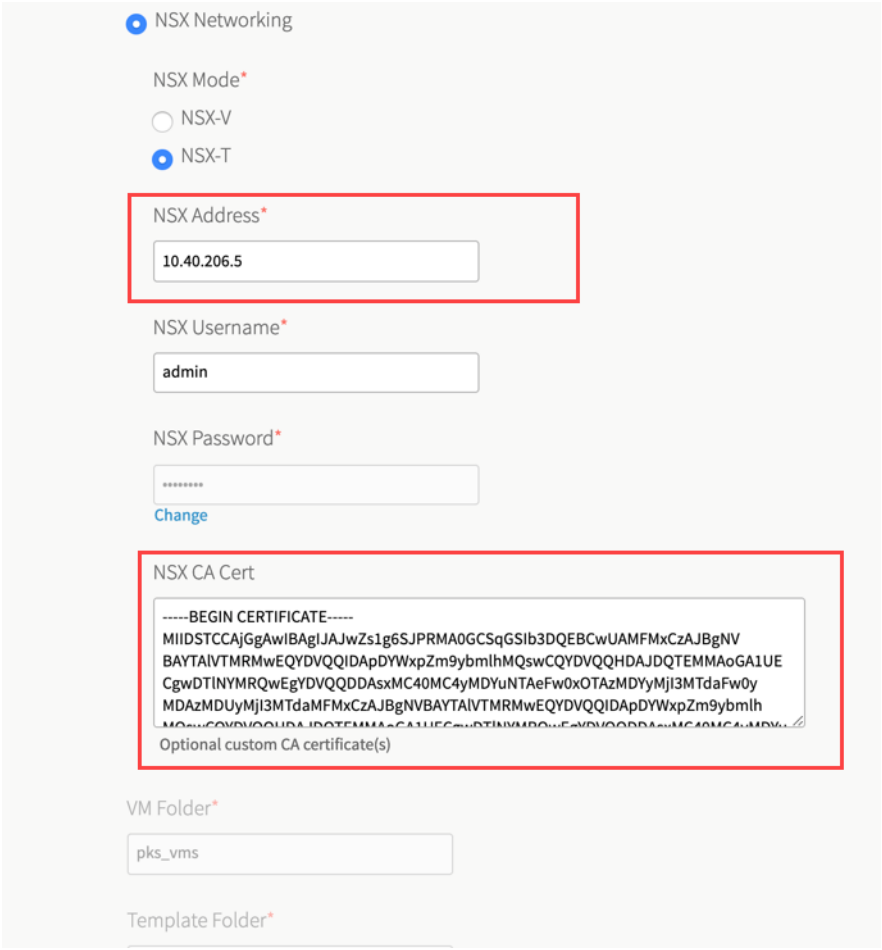
```
get certificate api
get certificate cluster
```

Step 7: Update PKS and BOSH with New NSX Manager Cert and VIP

The last procedure in the upgrade process is to modify the BOSH Tile and the PKS Tile with the new VIP address for the NSX Manager and the new NSX-T Manager CA cert (using VIP info). Apply the changes and ensure that the **Upgrade all clusters errand** is selected, then deploy PKS.

To update the BOSH tile:

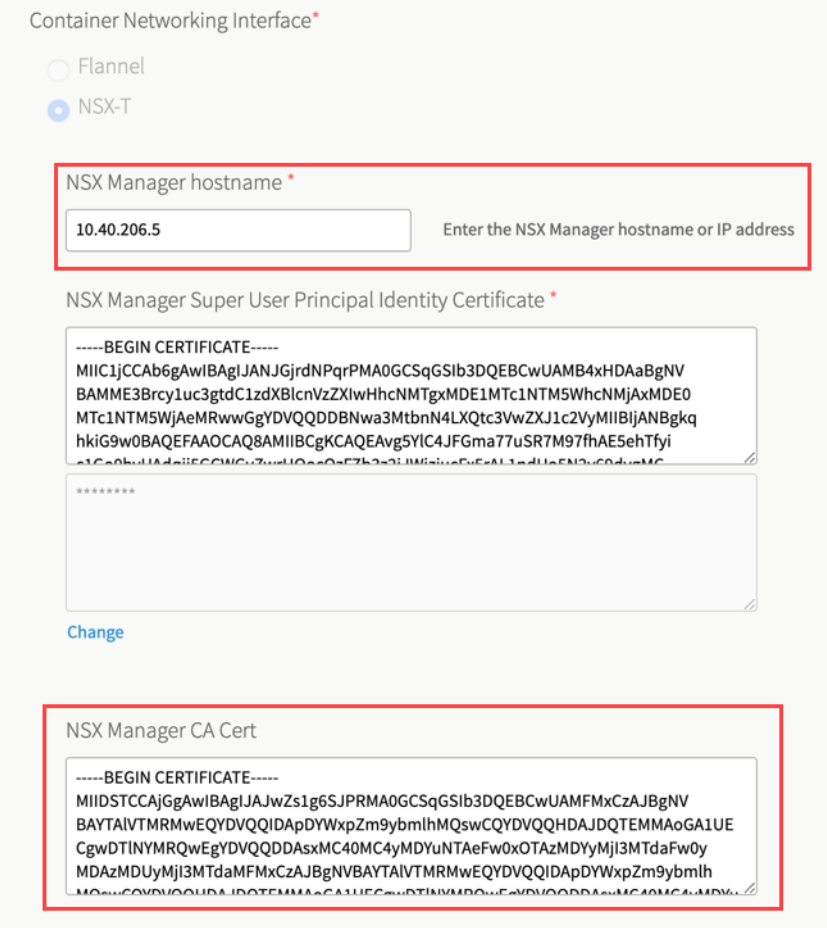
1. Log into Ops Manager.
2. In the BOSH Director tile, select the **vCenter Configuration** tab.
3. In the **NSX Address** field, enter the VIP address for the NSX Management Cluster.
4. In the **NSX CA Cert** field, enter the new CA certificate for the NSX Management Cluster that uses the VIP address.



5. Save the BOSH tile changes.

To update the PKS tile:

1. Log into Ops Manager.
2. In the PKS tile, select the **Networking** tab.
3. In the **NSX Manager hostname** field, enter the VIP address for the NSX Management Cluster.
4. In the **NSX Manager CA Cert** field, enter the new CA certificate for the NSX Management Cluster (that uses the VIP address).



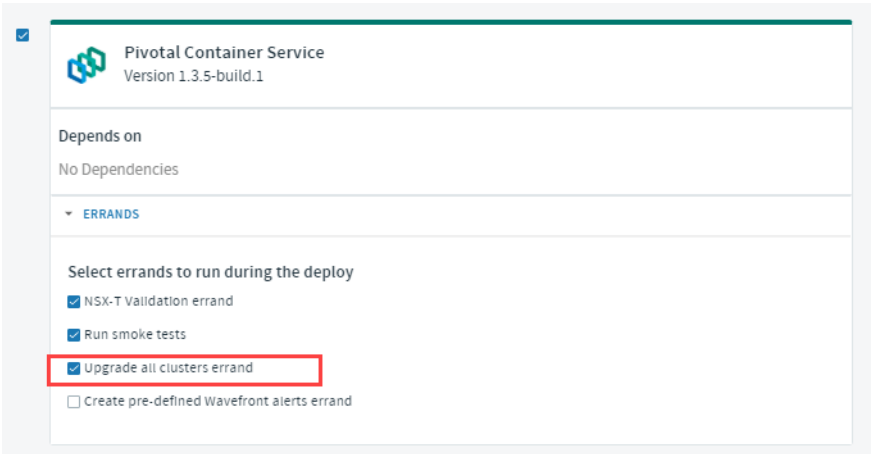
5. Save the PKS tile changes.

Step 8: Upgrade all Kubernetes Clusters

Once you have updated the PKS and BOSH tiles, apply the changes. Be sure to run the “Upgrade all [Kubernetes] clusters errand”. Doing so will allow NCP configurations on all Kubernetes clusters to be updated with the new NSX-T Management Cluster VIP and CA certificate.

To complete the upgrade:

1. Go to the **Installation Dashboard** in Ops Manager.
2. Click **Review Pending Changes.**
3. Expand the **Errands** list for PKS.
4. Ensure that the **Upgrade all clusters errand** is selected.



5. Click **Apply Changes**.

Step 9: Verify PKS Upgrade

Once the upgrade is complete, verify that NCP configuration is automatically updated with the new VIP (instead of individual NSX-T Manager node IP address).

1. To verify the NCP configuration has been updated with the new VIP, run a command for each Kubernetes cluster (service-instance_UUID):

   ```
   bosh ssh master/0 -d SERVICE-INSTANCE-UUID
   ```

   Where `SERVICE-INSTANCE-UUID` is the Kubernetes cluster UUID.

   For example:

   ```
   $ bosh ssh master/0 -d service-instance_d9b662d0-23e1-4239-b641-ed20ee62e692
   ```

   The returned “nsx_api_managers” address should be the new VIP address.

Step 10: Update PKS and Kubernetes CLIs

Update the PKS and Kubernetes CLIs on any local machine where you run commands that interact with your upgraded version of PKS.

To update your CLIs, download and re-install the PKS and Kubernetes CLI distributions that are provided with PKS on Pivotal Network.

For more information about installing the CLIs, see the following topics:

- Installing the PKS CLI
- Installing the Kubernetes CLI

Please send any feedback you have to pks-feedback@pivotal.io.

Maintaining Workload Uptime

Page last updated:

This topic describes how you can maintain workload uptime for Kubernetes clusters deployed with Pivotal Container Service (PKS).

To maintain workload uptime, configure the following settings in your deployment manifest:

1. Configure workload replicas to handle traffic during rolling upgrades.
2. Define an anti-affinity rule to evenly distribute workloads across the cluster.

To increase uptime, you can also refer to the documentation for the services that run on your clusters, and configure your workload based on the recommendations of the software vendor.

About Workload Upgrades

The PKS tile contains an errand that upgrades all Kubernetes clusters. Upgrades run on a single VM at a time. While one worker VM runs an upgrade, the workload on that VM goes down. The additional worker VMs continue to run replicas of your workload, maintaining the uptime of your workload.

Note: Ensure that your pods are bound to a *ReplicaSet* or *Deployment*. Naked pods are not rescheduled in the event of a node failure. For more information, see Configuration Best Practices in the Kubernetes documentation.

To prevent workload downtime during a cluster upgrade, Pivotal recommends running your workload on at least three worker VMs and using multiple replicas of your workloads spread across those VMs. You must edit your manifest to define the replica set and configure an anti-affinity rule to ensure that the replicas run on separate worker nodes.

Set Workload Replicas

Set the number of workload replicas to handle traffic during rolling upgrades. To replicate your workload on additional worker VMs, deploy the workload using a replica set.

Edit the `spec.replicas` value in your deployment manifest:

```
kind: Deployment
metadata:
  # ...
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: APP-NAME
```

See the following table for more information about this section of the manifest:

Key-Value Pair	Description
`spec:` `  replicas: 3`	Set this value to at least 3 to have at least three instances of your workload running at any time.
`app: APP-NAME`	Use this app name when you define the anti-affinity rule later in the spec.

Define an Anti-Affinity Rule

To distribute your workload across multiple worker VMs, you must use anti-affinity rules. If you do not define an anti-affinity rule, the replicated pods can be assigned to the same worker node. See the Kubernetes documentation for more information about anti-affinity rules.

To define an anti-affinity rule, add the `spec.template.spec.affinity` section to your deployment manifest:

```
kind: Deployment
metadata:
  # ...
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: APP-NAME
    spec:
      containers:
      - name: MY-APP
        image: MY-IMAGE
        ports:
        - containerPort: 12345
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            - labelSelector:
                matchExpressions:
                  - key: "app"
                    operator: In
                    values:
                    - APP-NAME
              topologyKey: "kubernetes.io/hostname"
```

See the following table for more information:

Key-Value Pair	Description
`podAntiAffinity: requiredDuringSchedulingIgnoredDuringExecution`	• When you set `podAntiAffinity` to the `requiredDuringSchedulingIgnoredDuringExecution` value, the pod is eligible to be scheduled only on worker nodes that are not running a replica of this pod. If the requirement cannot be met, scheduling fails. • Alternatively, you can set `podAntiAffinity` to the `preferredDuringSchedulingIgnoredDuringExecution` value. With this rule, the scheduler tries to schedule pod replicas on different worker nodes. If it is not possible, the scheduler assigns more than one pod to the same worker node.
`matchExpressions: - key: "app"`	This value matches `spec.template.metadata.labels.app`.
`values: - APP-NAME`	This value matches the `APP-NAME` you defined earlier in the spec.

Multi-AZ Worker

Kubernetes evenly spreads pods in a replication controller over multiple Availability Zones (AZs). For more granular control over scheduling pods, add an `Anti-Affinity Rule` to the deployment spec by replacing `"kubernetes.io/hostname"` with `"failure-domain.beta.kubernetes.io/zone"`.
For more information on scheduling pods, see Advanced Scheduling in Kubernetes on the Kubernetes Blog.

PersistentVolumes

If an AZ goes down, PersistentVolumes (PVs) and their data also go down and cannot be automatically re-attached. To preserve your PV data in the event of a fallen AZ, your persistent workload needs to have a failover mechanism in place.

Depending on the underlying storage type, PVs are either completely free of zonal information or can have multiple AZ labels attached. Both options enable a PV to travel between AZs.

To ensure the uptime of your PVs during a cluster upgrade, Pivotal recommends that you have at least two nodes per AZ. By configuring your workload as suggested, Kubernetes reschedules pods in the other node of the same AZ while BOSH is performing the upgrade.

For information about configuring PVs in PKS, see Configuring and Using PersistentVolumes.

For information about the supported storage topologies for PKS on vSphere, see PersistentVolume Storage Options on vSphere.

Please send any feedback you have to pks-feedback@pivotal.io.

Configuring the Upgrade Pipeline

Page last updated:

This topic describes how to set up a Concourse pipeline to perform automatic upgrades of a Pivotal Container Service (PKS) installation.

When you configure the upgrade pipeline, the pipeline upgrades your installation when a new PKS release becomes available on Pivotal Network.

By default, the pipeline upgrades when a new major patch version is available.

For more information about configuring and using Concourse for continuous integration (CI), see the Concourse documentation.

Download the Upgrade Pipeline

Perform the following steps:

1. From a browser, log in to Pivotal Network.
2. Navigate to the **PCF Platform Automation with Concourse** product page to download the upgrade-tile pipeline.

   Note: If you cannot access PCF Platform Automation with Concourse on Pivotal Network, contact Pivotal Support.

3. (Optional) Edit params.yml to configure the pipeline.
   - For example, edit the `product_version_regex` value to follow minor version updates.
4. Set the pipeline using the `fly` CLI for Concourse. See the upgrade-tile pipeline documentation for more information.

Please send any feedback you have to pks-feedback@pivotal.io.

Managing PKS

Page last updated:

This section describes how to manage Pivotal Container Service (PKS). See the following topics:

Please send any feedback you have to pks-feedback@pivotal.io.

Configuring PKS API Access

Page last updated:

This topic describes how to configure access to the Pivotal Container Service (PKS) API. See PKS API Authentication for more information about how the PKS API and UAA interact with your PKS deployment.

Configure Access to the PKS API

1. Locate your Ops Manager root CA certificate.
    - If Ops Manager generated your certificate, refer to the Retrieve the Ops Manager Root Certificate section of *Managing Certificates with the Ops Manager API*.
    - If you provided your own certificate, copy and paste the certificate you entered in the **PKS API** pane into a file.
2. Target your UAA server by running the following command:

    ```
    uaac target https\://PKS-API:8443 --ca-cert ROOT-CA-FILENAME
    ```

    Where:

    - `PKS-API` is the fully qualified domain name (FQDN) you use to access the PKS API. You configured this URL in the PKS API section of *Installing PKS* for your IaaS. For example, see Installing PKS on vSphere.
    - `ROOT-CA-FILENAME` is the path for the certificate file you downloaded in a previous step. For example:

        ```
        $ uaac target api.pks.example.com:8443 --ca-cert my-cert.cert
        ```

        Including `https://` in the PKS API URL is optional.
3. To request a token from the UAA server run the following command:

    ```
    uaac token client get admin -s UAA-ADMIN-SECRET`
    ```

    Where `UAA-ADMIN-SECRET` is your UAA admin secret. Refer to **Ops Manager** > **Pivotal Container Service** > **Credentials** > **Pks Uaa Management Admin Client** to retrieve your UAA admin secret.
4. Grant cluster access to new or existing users with UAA. For more information on granting cluster access to users or creating users, see the Grant PKS Access to a User section of *Managing Users in PKS with UAA*.

Log in to the PKS CLI as a User

For information about logging in to the PKS CLI as a user, see Logging in to PKS.

Note: If you are creating a test environment, you can log in to the PKS CLI without creating a PKS CLI-specific user account. Instead, you can use the existing Admin account and its UAA password to log in to the PKS CLI. Refer to **Ops Manager** > **Pivotal Container Service** > **Credentials** > **Uaa Admin Password** to retrieve your UAA Admin password and then follow the log in steps in Logging in to PKS.

Log in to PKS as an Automated Client

On the command line, run the following command to log in to the PKS CLI as an automated client for a script or service:

```
pks login -a PKS-API --client-name CLIENT-NAME --client-secret CLIENT-SECRET --ca-cert CERTIFICATE-PATH
```

Where:

- `PKS-API` is the domain name for the PKS API that you entered in **Ops Manager** > **Pivotal Container Service** > **PKS API** > **API Hostname (FQDN)**. For example, `api.pks.example.com`.
- `CLIENT-NAME` is your OAuth client ID.

- `CLIENT-SECRET` is your OAuth client secret.
- `CERTIFICATE-PATH` is the path to your root CA certificate. Provide the certificate to validate the PKS API certificate with SSL.
  For example:

```
$ pks login -a api.pks.example.com \
--client-name automated-client \
--client-secret randomly-generated-secret \
--ca-cert /var/tempest/workspaces/default/root_ca_certificate
```

Please send any feedback you have to pks-feedback@pivotal.io.

Creating and Configuring Load Balancers for PKS Clusters

Page last updated:

This section describes how to create and configure load balancers for Pivotal Container Service (PKS) clusters. See the following topics:

- Creating and Configuring a GCP Load Balancer for PKS Clusters
- Creating and Configuring an AWS Load Balancer for PKS Clusters
- Creating and Configuring an Azure Load Balancer for PKS Clusters

---

Please send any feedback you have to pks-feedback@pivotal.io.

Creating and Configuring a GCP Load Balancer for PKS Clusters

Page last updated:

This topic describes how to configure a Google Cloud Platform (GCP) load balancer for a Kubernetes cluster deployed by Pivotal Container Service (PKS).

Overview

A load balancer is a third-party device that distributes network and application traffic across resources. You can use a load balancer to access a PKS-deployed cluster from outside the network using the PKS API and `kubectl`. Using a load balancer can also prevent individual network components from being overloaded by high traffic.

You can configure GCP load balancers only for PKS clusters that are deployed on GCP.

To configure a GCP load balancer, follow the procedures below:

1. Create a GCP Load Balancer
2. Create a DNS Entry
3. Create the Cluster
4. Configure Load Balancer Back End
5. Create a Network Tag
6. Create Firewall Rules
7. Access the Cluster

To reconfigure a cluster load balancer, follow the procedures in Reconfigure Load Balancer below.

Prerequisites

The procedures in this topic have the following prerequisites:

- To complete these procedures, you must have already configured a load balancer to access the PKS API. For more information, see Creating a GCP Load Balancer for the PKS API.
- The version of the PKS CLI you are using must match the version of the PKS tile you are installing.

Configure GCP Load Balancer

Follow the procedures in this section to create and configure a load balancer for PKS-deployed Kubernetes clusters using GCP. Modify the example commands in these procedures to match your PKS installation.

Create a GCP Load Balancer

To create a GCP load balancer for your PKS clusters, do the following:

1. Navigate to the Google Cloud Platform console.
2. In the sidebar menu, select **Network Services** > **Load balancing**.
3. Click **Create a Load Balancer**.
4. In the **TCP Load Balancing** pane, click **Start configuration**.
5. Click **Continue**. The **New TCP load balancer** menu opens.

6. Give the load balancer a name. For example, `my-cluster`.
7. Click **Frontend configuration** and configure the following settings:
    a. Click **IP**.
    b. Select **Create IP address**.
    c. Give the IP address a name. For example, `my-cluster-ip`.
    d. Click **Reserve**. GCP assigns an IP address.
    e. In the **Port** field, enter `8443`.
    f. Click **Done** to complete front end configuration.
8. Review your load balancer configuration and click **Create**.

Create a DNS Entry

To create a DNS entry in GCP for your PKS cluster, do the following:

1. From the GCP console, navigate to **Network Services > Cloud DNS**.
2. Select the DNS zone for your domain. To retrieve your zone name, do one of the following:
    - **If you installed PKS manually:** Select the zone you used when you created the PKS API DNS entry. See the Create a DNS Entry section in *Creating a GCP Load Balancer for the PKS API*.
    - **If you installed PKS using Terraform:** Run `terraform output` and locate the value for `dns_managed_zone`.
3. Click **Add record set**.
4. Under **DNS Name**, enter a subdomain for the load balancer. For example, if your domain is `example.com`, enter `my-cluster` in this field to use `my-cluster.example.com` as your PKS cluster hostname.
5. Under **Resource Record Type**, select **A** to create a DNS address record.
6. Enter a value for **TTL** and select a **TTL Unit**.
7. Enter the GCP-assigned IP address you created in Create a Load Balancer above.
8. Click **Create**.

Create the Cluster

To create a cluster, follow the steps input Create a Kubernetes Cluster section of *Creating Clusters*. Use the PKS cluster hostname from the above step as the external hostname when you run the `pks create-cluster` command.

Configure Load Balancer Back End

To configure the back end of the load balancer, do the following:

1. Record the ID for your master node VMs by doing one of the following:
    - Complete Identify Kubernetes Cluster Master VMs in *Creating Clusters*
    - Complete the following procedure:
        1. Log in to PKS by running the following command:

            ```
            pks login -a PKS-API -u USERNAME -k
            ```

            Where:

            - `PKS-API` is the domain name for the PKS API that you entered in **Ops Manager** > **Enterprise PKS** > **PKS API** > **API Hostname (FQDN)**. For example, `api.pks.example.com`.
            - `USERNAME` is your user name.
        2. Locate the master node IP addresses by running the following command:

```
pks cluster CLUSTER-NAME
```

Where `CLUSTER-NAME` is the unique name for your cluster.

From the output of this command, record the value of **Kubernetes Master IP(s)**. This value lists the IP addresses of all master node VMs in the cluster.

3. Navigate to the Google Cloud Platform console.
4. From the sidebar menu, navigate to **Compute Engine** > **VM instances**.
5. Filter the VMs using the network name you provided when you deployed Ops Manager on GCP.
6. Record the IDs of the master node VMs associated with the IP addresses you recorded in the above step. The above IP addresses appear under the **Internal IP** column.

2. In the Google Cloud Platform console, from the sidebar menu, navigate to **Network Services** > **Load balancing**.
3. Select the load balancer you created for the cluster and click **Edit**.
4. Click **Backend configuration** and configure the following settings:
   a. Select all the master node VMs for your cluster from the dropdown.

> **Warning:** If master VMs are recreated for any reason, such as a stemcell upgrade, you must reconfigure the load balancer to target the new master VMs. For more information, see the Reconfigure Load Balancer section below.

   b. Specify any other configuration options you require and click **Update** to complete back end configuration.

> **Note:** For clusters with multiple master node VMs, health checks on port 8443 are recommended.

Create a Network Tag

To create a network tag, do the following:

1. In the Google Cloud Platform sidebar menu, select **Compute Engine** > **VM instances**.
2. Filter to find the master instances of your cluster. Type `master` in the **Filter VM Instances** search box and press **Enter**.
3. Click the name of the master instances. The **VM instance details** menu opens.
4. Click **Edit**.
5. Click in the **Network tags** field and type a human-readable name in lowercase letters. Press **Enter** to create the network tag.
6. Scroll to the bottom of the screen and click **Save**.

Create Firewall Rules

To create firewall rules, do the following:

1. In the Google Cloud Platform sidebar menu, select **VPC Network** > **Firewall Rules**.
2. Click **Create Firewall Rule**. The **Create a firewall rule** menu opens.
3. Give your firewall rule a human-readable name in lower case letters. For ease of use, you may want to align this name with the name of the load balancer you created in Create a GCP Load Balancer.
4. In the **Network** menu, select the VPC network on which you have deployed the PKS tile.
5. In the **Direction of traffic** field, select **Ingress**.
6. In the **Action on match** field, select **Allow**.
7. Confirm that the **Targets** menu is set to `Specified target tags` and enter the tag you made in Create a Network Tag in the **Target tags** field.
8. In the **Source filter** field, choose an option to filter source traffic.

9. Based on your choice in the **Source filter** field, specify IP addresses, Subnets, or Source tags to allow access to your cluster.
10. In the **Protocols and ports** field, choose **Specified protocols and ports** and enter the port number you specified in Create a GCP Load Balancer, prepended by `tcp:`. For example: `tcp:8443`.
11. Specify any other configuration options you require and click **Done** to complete front end configuration.
12. Click **Create**.

Access the Cluster

To complete cluster configuration, do the following:

1. From your local workstation, run `pks get-credentials CLUSTER-NAME`. This command creates a local `kubeconfig` that allows you to manage the cluster. For more information about the `pks get-credentials` command, see Retrieving Cluster Credentials and Configuration below.
2. Run `kubectl cluster-info` to confirm you can access your cluster using the Kubernetes CLI.

See Managing PKS for information about checking cluster health and viewing cluster logs.

Reconfigure Load Balancer

If Kubernetes master node VMs are recreated for any reason, you must reconfigure your cluster load balancers to point to the new master VMs. For example, after a stemcell upgrade, BOSH recreates the VMs in your deployment.

To reconfigure your GCP cluster load balancer to use the new master VMs, do the following:

1. Locate the VM IDs of the new master node VMs for the cluster. For information about locating the VM IDs, see Identify Kubernetes Cluster Master VMs in *Creating Clusters*.
2. Navigate to the GCP console.
3. In the sidebar menu, select **Network Services > Load balancing**.
4. Select your cluster load balancer and click **Edit**.
5. Click **Backend configuration**.
6. Click **Select existing instances**.
7. Select the new master VM IDs from the dropdown. Use the VM IDs you located in the first step of this procedure.
8. Click **Update**.

Creating and Configuring an AWS Load Balancer for PKS Clusters

This topic describes how to configure a Amazon Web Services (AWS) load balancer for your Pivotal Container Service (PKS) cluster.

A load balancer is a third-party device that distributes network and application traffic across resources. Using a load balancer can also prevent individual network components from being overloaded by high traffic. For more information about the different types of load balancers used in a PKS deployment see Load Balancers in PKS.

You can use an AWS PKS cluster load balancer to secure and facilitate access to a PKS cluster from outside the network. You can also reconfigure your AWS PKS cluster load balancers.

Using an AWS PKS cluster load balancer is optional, but adding one to your Kubernetes cluster can make it easier to manage the cluster using the PKS API and `kubectl`.

Note: If Kubernetes master node VMs are recreated for any reason, you must reconfigure your AWS PKS cluster load balancers to point to the new master VMs.

Prerequisite

The version of the PKS CLI you are using must match the version of the PKS tile you are installing.

Note: This procedure uses example commands which you should modify to represent the details of your PKS installation.

Configure AWS Load Balancer

Step 1: Define Load Balancer

To define your load balancer using AWS, you must provide a name, select a VPC, specify listeners, and select subnets where you want to create the load balancer.

Perform the following steps:

1. In a browser, navigate to the AWS Management Console.
2. Under **Compute**, click **EC2**.
3. In the **EC2 Dashboard**, under **Load Balancing**, click **Load Balancers**.
4. Click **Create Load Balancer**.
5. Under **Classic Load Balancer**, click **Create**.
6. On the **Define Load Balancer** page, complete the **Basic Configuration** section as follows:
7. **Load Balancer name**: Name the load balancer. Pivotal recommends that you name your load balancer `k8s-master-CLUSTERNAME` where `CLUSTERNAME` is a unique name that you provide when creating the cluster. For example, `k8s-master-mycluster`.
    a. **Create LB inside**: Select the VPC where you installed Ops Manager.
    b. **Create an internal load balancer**: Do not enable this checkbox. The cluster load balancer must be internet-facing.
8. Complete the **Listeners Configuration** section as follows:
    a. Configure the first listener as follows.
        - Under **Load Balancer Protocol**, select **TCP**.
        - Under **Load Balancer Port**, enter `8443`.
        - Under **Instance Protocol**, select **TCP**.
        - Under **Instance Port**, enter `8443`.

9. Under **Select Subnets**, select the public subnets for your load balancer in the availability zones where you want to create the load balancer.
10. Click **Next: Assign Security Groups**.

Step 2: Assign Security Groups

Perform the following steps to assign security groups:

1. On the **Assign Security Groups** page, select one of the following:
    - **Create a new security group**: Complete the security group configuration as follows:
        1. **Security group name**: Name your security group.
        2. Confirm that your security group includes **Protocol** `TCP` with **Ports** `8443`.
    - **Select an existing security group**: Select the default security group. The default security group includes includes **Protocol** `TCP` with **Ports** `8443`.
2. Click **Next: Configure Security Settings**.

Step 3: Configure Security Settings

On the **Configure Security Settings** page, ignore the warning. SSL termination is done on the Kubernetes API.

Step 4: Configure Health Check

Perform the following steps to configure the health check:

1. On the **Configure Health Check** page, set the **Ping Protocol** to `TCP`.
2. For **Ping Port**, enter `8443`.
3. Click **Next: Add EC2 Instances**.

Step 5: Add EC2 Instances

Perform the following steps:

1. Verify the settings under **Availability Zone Distribution**.
2. Click **Add Tags**.

(Optional) Step 6: Add Tags

Perform the following steps to add tags:

1. Add tags to your resources to help organize and identify them. Each tag consists of a case-sensitive key-value pair.
2. Click **Review and Create**.

Step 7: Review and Create the Load Balancer

Perform the following steps to review your load balancer details and create your load balancer:

1. On the **Review** page, review your load balancer details and edit any as necessary.
2. Click **Create**.

Step 8: Create a Cluster

Create a Kubernetes cluster using the AWS-assigned address of your load balancer as the external hostname when you run the `pks create-cluster` command.

For example:

```
$ pks create-cluster my-cluster \
  --external-hostname example111a6511e9a099028c856be95-155233362.eu-west-1.elb.amazonaws.com \
  --plan small --num-nodes 10
```

For more information, see Create a Kubernetes Cluster section of *Creating Clusters*.

Step 9: Point the Load Balancer to All Master VMs

1. Locate the VM IDs of all master node VMs for your cluster. For information about locating the VM IDs, see Identify Kubernetes Cluster Master VMs in *Creating Clusters*.
2. Navigate to the AWS console.
3. Under EC2, select **Load balancers**.
4. Select the load balancer.
5. On the **Instances** tab, click **Edit instances**.
6. Select all master nodes in the list of VMs.
7. Click **Save**.

Reconfigure AWS Load Balancer

If Kubernetes master node VMs are recreated for any reason, you must reconfigure your cluster load balancers to point to the new master VMs. For example, after a stemcell upgrade, BOSH recreates the VMs in your deployment.

To reconfigure your AWS cluster load balancer to use the new master VMs, do the following:

1. Locate the VM IDs of the new master node VMs for the cluster. For information about locating the VM IDs, see Identify Kubernetes Cluster Master VMs in *Creating Clusters*.
2. Navigate to the AWS console.
3. Under EC2, select **Load balancers**.
4. Select the load balancer.
5. On the **Instances** tab, click **Edit instances**.
6. Select the new master nodes in the list of VMs.
7. Click **Save**.

Creating and Configuring an Azure Load Balancer for PKS Clusters

Page last updated:

This topic describes how to create and configure an Azure load balancer for your Pivotal Container Service (PKS) cluster. Using an Azure load balancer is optional, but you may want to add one to your Kubernetes cluster to manage the cluster using the PKS API and Kubernetes CLI (kubectl).

A load balancer is a third-party device that distributes network and application traffic across resources. You can use a load balancer to secure and facilitate access to a PKS cluster from outside the network. Using a load balancer can also prevent individual network components from being overloaded by high traffic.

Note: If your Kubernetes master node VMs are recreated for any reason, you must reconfigure your cluster load balancers to point to the new master VMs. For instructions, see Reconfigure Load Balancer.

Prerequisites

To complete the steps below, you must identify the PKS API virtual machine (VM). You can find the name in the following ways:

- In the Azure Dashboard, locate the VM tagged with `instance_group:pivotal-container-service`.
- On the command line, run `bosh vms`.

Create and Configure a Load Balancer

Follow the steps below to create and configure an Azure load balancer for your PKS cluster.

Create Load Balancer

1. In a browser, navigate to the Azure Dashboard.
2. Open the **Load Balancers** service.
3. Click **Add**.
4. On the **Create load balancer** page, complete the form as follows:
    a. **Name**: Name the load balancer.
    b. **Type**: Select **Public**.
    c. **SKU**: Select **Standard**.
    d. **Public IP address**: Select **Create new** and name the new IP address.
    e. **Availability zone**: Select an availability zone or **Zone-redundant**.
    f. **Subscription**: Select the subscription which has PKS deployed.
    g. **Resource group**: Select the resource group which has PKS deployed.
    h. **Location**: Select the location group which has PKS deployed.
5. Click **Create**.

Create Backend Pool

1. From the Azure Dashboard, open the **Load Balancers** service.
2. Click the name of the load balancer that you created in Create Load Balancer.
3. On your load balancer page, locate and record the IP address of your load balancer.
4. In the **Settings** menu, select **Backend pools**.
5. On the **Backend pools** page, click **Add**.

6. On the **Add backend pool** page, complete the form as follows:

    a. **Name**: Name the backend pool.
    b. **Virtual network**: Select the virtual network where the PKS API VM is deployed.
    c. **Virtual machine**: Select all of the master VMs for your cluster. For information about identifying the master VM IDs, see Identify Kubernetes Cluster Master VMs in *Creating Clusters*.

7. Click **Add**.

Create Health Probe

1. From the Azure Dashboard, open the **Load Balancers** service.

2. In the **Settings** menu, select **Health probes**.

3. On the **Health probes** page, click **Add**.

4. On the **Add health probe** page, complete the form as follows:

    a. **Name**: Name the health probe.
    b. **Protocol**: Select **TCP**.
    c. **Port**: Enter `8443`.
    d. **Interval**: Enter the interval of time to wait between probe attempts.
    e. **Unhealthy Threshold**: Enter a number of consecutive probe failures that must occur before a VM is considered unhealthy.

5. Click **OK**.

Create Load Balancing Rule

1. From the Azure Dashboard, open the **Load Balancers** service.

2. In the **Settings** menu, select **Load Balancing Rules**.

3. On the **Load balancing rules** page, click **Add**.

4. On the **Add load balancing rules** page, complete the form as follows:

    a. **Name**: Name the load balancing rule.
    b. **IP Version**: Select **IPv4**.
    c. **Frontend IP address**: Select the appropriate IP address. Clients communicate with your load balancer on the selected IP address and service traffic is routed to the target VM by this NAT rule.
    d. **Protocol**: Select **TCP**.
    e. **Port**: Enter `8443`.
    f. **Backend port**: Enter `8443`.
    g. **Backend Pool**: Select the backend pool that you created in Create Backend Pool.
    h. **Health Probe**: Select the health probe that you created in Create Health Probe.
    i. **Session persistence**: Select **None**.

5. Click **OK**.

Create Inbound Security Rule

1. From the Azure Dashboard, open the **Security Groups** service.

2. Click the name of the Security Group attached to the subnet where PKS API is deployed. If you deployed PKS using Terrform, the name of the Security Group ends with the suffix `bosh-deployed-vms-security-group`.

3. In the **Settings** menu for your security group, select **Inbound security rules**.

4. Click **Add**.

5. On the **Add inbound security rule** page, click **Advanced** and complete the form as follows:

    a. **Name**: Name the inbound security rule.

b. **Source**: Select **Any**.
c. **Source port range**: Enter `*`.
d. **Destination**: Select **Any**.
e. **Destination port range**: Enter `8443`.

6. Click **OK**.

Verify Hostname Resolution

Verify that the **External hostname** used when creating a Kubernetes cluster resolves to the IP address of the load balancer.

For more information, see Create a Kubernetes Cluster in *Creating Clusters*.

Reconfigure Load Balancer

If your Kubernetes master node VMs are recreated for any reason, you must reconfigure your cluster load balancers to point to the new master VMs. For example, after a stemcell upgrade, BOSH recreates the VMs in your deployment.

To reconfigure your Azure cluster load balancer to use the new master VMs, do the following:

1. Identify the VM IDs of the new master node VMs for the cluster. For information about identifying the master VM IDs, see Identify Kubernetes Cluster Master VMs in *Creating Clusters*.
2. In a browser, navigate to the Azure Dashboard.
3. Open the **Load Balancers** service.
4. Select the load balancer for your cluster.
5. In the **Settings** menu, select **Backend pools**.
6. Update the VMs list with the new master VM IDs.
7. Click **Save**.

Please send any feedback you have to pks-feedback@pivotal.io.

Managing Users in PKS with UAA

Page last updated:

This topic describes how to manage users in Pivotal Container Service (PKS) with User Account and Authentication (UAA). Create and manage users in UAA with the UAA Command Line Interface (UAAC).

How to Use UAAC

Use the UAA Command Line Interface (UAAC) to interact with the UAA server. You can either run UAAC commands from the Ops Manager VM or install UAAC on your local workstation.

To run UAAC commands from the Ops Manager VM, see the following SSH procedures for vSphere or Google Cloud Platform (GCP).

To install UAAC locally, see Component: User Account and Authentication (UAA) Server.

SSH into the Ops Manager VM on vSphere

To SSH into the Ops Manager VM on vSphere, you need the credentials used to import the PCF .ova or .ovf file into your virtualization system. You set these credentials when you installed Ops Manager.

Note: If you lose your credentials, you must shut down the Ops Manager VM in the vSphere UI and reset the password. See vCenter Password Requirements and Lockout Behavior in the vSphere documentation for more information.

1. From a command line, run the following command to SSH into the Ops Manager VM:

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

   Where `OPS-MANAGER-FQDN` is the fully qualified domain name (FQDN) of Ops Manager.

2. When prompted, enter the password that you set during the .ova deployment into vCenter. For example:

   ```
   $ ssh ubuntu@my-opsmanager-fqdn.example.com
   Password: **********
   ```

3. Proceed to the Log in as a UAA Admin section to manage users with UAAC.

SSH into the Ops Manager VM on GCP

To SSH into the Ops Manager VM in GCP, do the following:

1. Confirm that you have installed the gcloud CLI. See Downloading gcloud in the Google Cloud Platform documentation for more information.
2. From the GCP console, click **Compute Engine**.
3. Locate the Ops Manager VM in the **VM Instances** list.
4. Click the **SSH** menu button.
5. Copy the SSH command that appears in the popup window.
6. Paste the command into your terminal window to SSH to the Ops Manager VM. For example:

   ```
   $ gcloud compute ssh om-pcf-1a --zone us-central1-b
   ```

7. Run `sudo su - ubuntu` to switch to the `ubuntu` user.
8. Proceed to the Log in as a UAA Admin section to manage users with UAAC.

SSH into the Ops Manager VM on AWS

To SSH into the Ops Manager VM on AWS, you need the key pair you used when you created the Ops Manager VM. To see the name of the key pair, click on the Ops Manager VM in the AWS console and locate the `key pair name` in the properties.

To SSH into the Ops Manager VM on AWS, do the following:

1. From the AWS console, locate the Ops Manager fully qualified domain name on the AWS **EC2 instances** page.
2. Run `chmod 600 ops_mgr.pem` to change the permissions on the `.pem` file to be more restrictive. For example:

   ```
   $ chmod 600 ops_mgr.pem
   ```

3. Run the following command to SSH into the Ops Manager VM:

   ```
   ssh -i ops_mgr.pem ubuntu@OPS-MANAGER-FQDN
   ```

   Where `OPS-MANAGER-FQDN` is the fully qualified domain name of Ops Manager. For example:

   ```
   $ ssh -i ops_mgr.pem ubuntu@my-opsmanager-fqdn.example.com
   ```

4. Proceed to the Log in as a UAA Admin section to manage users with UAAC.

SSH into the Ops Manager VM on Azure

To SSH into the Ops Manager VM with SSH in Azure, you need the SSH key pair you used when you created the Ops Manager VM. If you need to reset the SSH key, locate the Ops Manager VM in the Azure portal and click **Reset Password**.

To log in to the Ops Manager VM with SSH in Azure, do the following:

1. From the Azure portal, locate the Ops Manager FQDN by selecting the VM.
2. Change the permissions for your SSH private key by running the following command:

   ```
   chmod 600 PRIVATE-KEY
   ```

   Where `PRIVATE-KEY` is the name of your SSH private key.

3. SSH into the Ops Manager VM by running the following command:

   ```
   ssh -i PRIVATE-KEY ubuntu@OPS-MANAGER-FQDN
   ```

   Where:

   - `OPS-MANAGER-FQDN` is FQDN of Ops Manager.
   - `PRIVATE-KEY` is the name of your SSH private key.

   For example:

   ```
   $ ssh -i id_rsa ubuntu@my-opsmanager-fqdn.example.com
   ```

4. Proceed to the Log in as a UAA Admin section to manage users with UAAC.

Log in as a UAA Admin

To retrieve the PKS UAA management admin client secret, do the following:

1. In a web browser, navigate to the fully qualified domain name of Ops Manager and click the **Pivotal Container Service** tile.
2. Click **Credentials**.
3. To view the secret, click **Link to Credential** next to **Pks Uaa Management Admin Client**. The client username is `admin`.

4. On the command line, run the following command to target your UAA server:

    ```
    uaac target https://PKS-API:8443 --ca-cert ROOT-CA-FILENAME
    ```

    Where:

    - `PKS-API` is the URL to your PKS API server. You configured this URL in the PKS API section of*Installing PKS* for your IaaS. For example, see Installing PKS on vSphere.
    - `ROOT-CA-FILENAME` is the certificate file you downloaded in Configuring PKS API Access. If you are logged in to the Ops Manager VM, the root certificate is located at `/var/tempest/workspaces/default/root_ca_certificate`. For example:

        ```
        $ uaac target api.pks.example.com:8443 --ca-cert /var/tempest/workspaces/default/root_ca_certificate
        ```

        Note: If you receive an `Unknown key: Max-Age = 86400` warning message, you can safely ignore it because it has no impact.

5. Run the following command to authenticate with UAA using the secret you retrieved in a previous step:

    ```
    uaac token client get admin -s ADMIN-CLIENT-SECRET
    ```

    Where `ADMIN-CLIENT-SECRET` is your PKS UAA management admin client secret.

Grant PKS Access

PKS access gives users the ability to deploy and manage Kubernetes clusters. As an Admin user, you can assign the following UAA scopes to users, external LDAP groups, and clients:

- `pks.clusters.manage`: Accounts with this scope can create and access their own clusters.
- `pks.clusters.admin`: Accounts with this scope can create and access all clusters.

Grant PKS Access to a User

You can create a new UAA user with PKS access by performing the following steps:

1. Log in as the UAA admin using the procedure in Log in as a UAA Admin.
2. To create a new user, run the following command:

    ```
    uaac user add USERNAME --emails USER-EMAIL -p USER-PASSWORD
    ```

    For example:

    ```
    $ uaac user add alana --emails alana@example.com -p password
    ```

3. Run the following command to assign a scope to the user to allow them to access Kubernetes clusters:

    ```
    uaac member add UAA-SCOPE USERNAME
    ```

    Where `UAA-SCOPE` is one of the UAA scopes defined in Grant PKS Access. For example:

    ```
    $ uaac member add pks.clusters.admin alana
    ```

Grant PKS Access to an External LDAP Group

Connecting PKS to a LDAP external user store allows the User Account and Authentication (UAA) server to delegate authentication to existing enterprise user stores.

Note: When integrating with an external identity provider such as LDAP, authentication within the UAA becomes chained. UAA first attempts to authenticate with a user's credentials against the UAA user store before the external provider, LDAP. For more information, see Chained

Authentication in the *User Account and Authentication LDAP Integration* GitHub documentation.

For more information about the process used by the UAA Server when it attempts to authenticate a user through LDAP, see the Configuring LDAP Integration with Pivotal Cloud Foundry Knowledge Base article.

To grant PKS access to an external LDAP group, perform the following steps:

1. Log in as the UAA admin using the procedure in Log in as a UAA Admin.
2. To assign the `pks.clusters.manage` scope to all users in an LDAP group, run the following command:

   ```
   uaac group map --name pks.clusters.manage GROUP-DISTINGUISHED-NAME
   ```

   Where `GROUP-DISTINGUISHED-NAME` is the LDAP Distinguished Name (DN) for the group. For example:

   ```
   $ uaac group map --name pks.clusters.manage cn=operators,ou=groups,dc=example,dc=com
   ```

   For more information about LDAP DNs, see the LDAP DNs and RDNs in the LDAP documentation.
3. (Optional) To assign the `pks.clusters.admin` scope to all users in an LDAP group, run the following command:

   ```
   uaac group map --name pks.clusters.admin GROUP-DISTINGUISHED-NAME
   ```

   Where `GROUP-DISTINGUISHED-NAME` is the LDAP DN for the group. For example:

   ```
   $ uaac group map --name pks.clusters.admin cn=operators,ou=groups,dc=example,dc=com
   ```

Grant PKS Access to a Client

To grant PKS access to an automated client for a script or service, perform the following steps:

1. Log in as the UAA admin using the procedure Log in as a UAA Admin.
2. Run the following command to create a client with the desired scopes:

   ```
   uaac client add CLIENT-NAME -s CLIENT-SECRET \
   --authorized_grant_types client_credentials \
   --authorities UAA-SCOPES
   ```

   Where:

   - `CLIENT-NAME` and `CLIENT-SECRET` are the client credentials.
   - `UAA-SCOPES` is fines one or more of the UAA scopes defined in Grant PKS Access , separated by a comma. For example:

     ```
     $ uaac client add automated-client \
     -s randomly-generated-secret
     --authorized_grant_types client_credentials  \
     --authorities pks.clusters.admin,pks.clusters.manage
     ```

Grant Cluster Access

You can grant a user or a group access to an entire cluster with a `ClusterRole` or to a namespace within a given cluster with a `Role`.

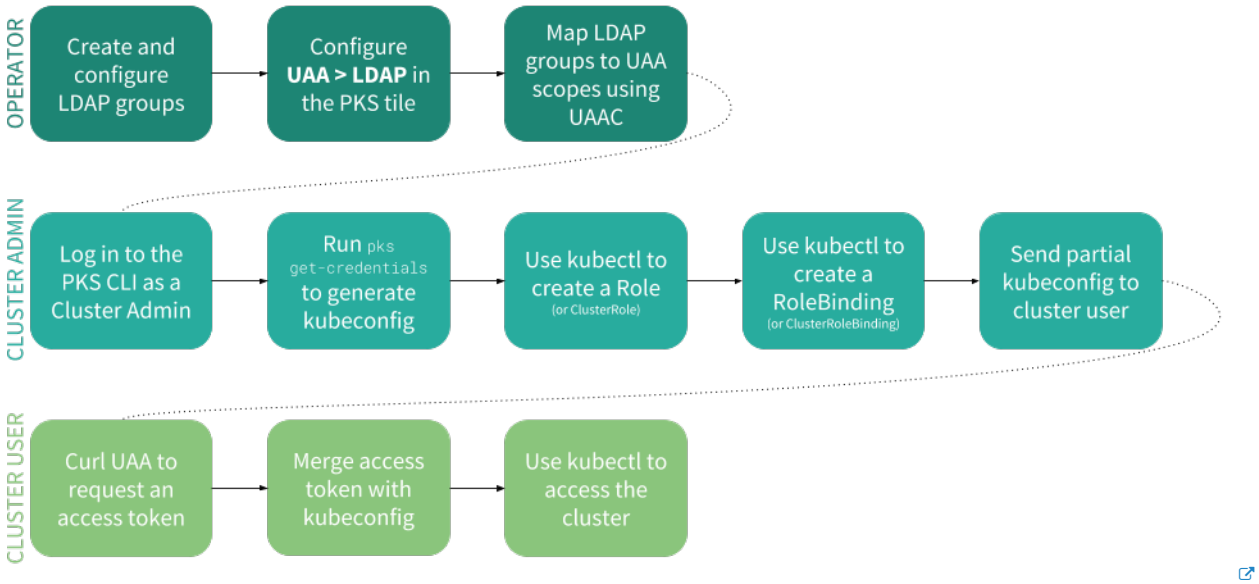
The admin of the cluster must then create a `ClusterRoleBinding` or a `RoleBinding` for that Kubernetes end user.

For more information, see RoleBinding and ClusterRoleBinding in the Kubernetes documentation.

Grant Cluster Access to a User

After being granted cluster access, the Kubernetes end user can use the Kubernetes Command Line Interface (kubectl) to connect to the cluster and perform actions as configured by their cluster admin. However, even with this access, Kubernetes end users cannot create, resize, or delete clusters.

The following diagram outlines the workflow you use to grant cluster access to a user who belongs to an LDAP group:



Note: In order for cluster admins to grant cluster access to Kubernetes end users, cluster admins must ensure that they have selected **Enable UAA as OIDC provider** in the UAA section of the PKS tile. Once you enable OIDC, you must run `get-credentials` again to update your existing kubeconfig.

To grant cluster access to other users, the cluster admin must perform the following actions:

1. Run the following command to log in to PKS client using LDAP credentials:

   ```
   pks login -u LDAP-NAME -p LDAP-PASSWORD -a PKS-API --ca-cert ROOT-CA-FILENAME
   ```

   Where:

   - `LDAP-USER-NAME` is the cluster admin's LDAP username.
   - `LDAP-PASSWORD` is the cluster admin's LDAP password.
   - `PKS-API` is the fully qualified domain name you use to access the PKS API.

2. Run the following command to confirm that you can successfully connect to a cluster and use kubectl as a cluster admin:

   ```
   pks get-credentials CLUSTER-NAME
   ```

   This step creates a `ClusterRoleBinding` for the LDAP cluster admin.

3. When prompted, re-enter your LDAP password.

4. Create a spec YML file with either the `Role` or `ClusterRole` for your Kubernetes end user.

   ```
   kind: ROLE-TYPE
   apiVersion: rbac.authorization.k8s.io/v1
   metadata:
     namespace: NAMESPACE
     name: ROLE-OR-CLUSTER-ROLE-NAME
   rules:
   - apiGroups:
     resources: RESOURCE
     verbs: API-REQUEST-VERB
   ```

   Where:

   - `ROLE-TYPE` is the type of role you are creating. This must be either `Role` or `ClusterRole`.
   - `NAMESPACE` is the namespace within the cluster. This is omitted when creating a `ClusterRole`.
   - `ROLE-OR-CLUSTER-ROLE-NAME` is the name of the `Role` or `ClusterRole` you are creating. This name is created by the cluster admin.
   - `RESOURCE` is the resource you are granting access to. It must be specified in a comma-separated array. An example resource could be `["pod-reader"]`.
   - `API-REQUEST-VERB` is used to specify resource requests. For more information, see Determine the Request Verb in the Kubernetes

documentation.

5. Run the following command to create the `Role` or `ClusterRole` resource based on your spec file:

```
kubectl create -f ROLE-SPEC.yml
```

6. Create a spec YML file containing either a `ClusterRoleBinding` or `RoleBinding` for the Kubernetes end user.

```
kind: ROLE-BINDING-TYPE
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: ROLE-OR-CLUSTER-ROLE-BINDING-NAME
  namespace: NAMESPACE
subjects:
- kind: User
  name: USERNAME
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ROLE-TYPE
  name: ROLE-OR-CLUSTER-ROLE-BINDING-NAME
  apiGroup: rbac.authorization.k8s.io
```

   Where:

   - `ROLE-BINDING-TYPE` is the type of role binding you are creating. This must be either `RoleBinding` or `ClusterRoleBinding`.
   - `ROLE-OR-CLUSTER-ROLE-BINDING-NAME` is the name of the role binding. This name is created by the cluster admin.
   - `NAMESPACE` is the namespace within the cluster. This is omitted when creating a `ClusterRole`.
   - `USERNAME` is the Kubernetes end user's username. If your organization uses LDAP, for example, this is your LDAP username.
   - `ROLE-TYPE` is the type of role you created in the previous step. This must be either `Role` or `ClusterRole`.
   - `ROLE-OR-CLUSTER-ROLE-NAME` is the name of the `Role` or `ClusterRole` you are creating.

7. Run the following command to create the above defined `ClusterRoleBinding` resource in the cluster:

```
kubectl apply -f ROLE-BINDING-SPEC.yml
```

8. The cluster admin partially completes the `kubeconfig` by detailing the following:

   - `clusters.cluster.certificate-authority-data`
   - `clusters.cluster.server`
   - `cluster.name`
   - `contexts.context.cluster`
   - `contexts.context.name`
   - `current-context`
   - `users.user.auth-provider.config.idp-issuer-url`

9. The cluster admin sends the partially completed `kubeconfig` to their Kubernetes end user. Review the example kubeconfig file below. For more information about organizing information using kubeconfig, see Organizing Cluster Access Using kubeconfig Files in the Kubernetes documentation.

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: PROVIDED-BY-ADMIN
    server: PROVIDED-BY-ADMIN
  name: PROVIDED-BY-ADMIN
contexts:
- context:
    cluster: PROVIDED-BY-ADMIN
    user: PROVIDED-BY-USER
  name:  PROVIDED-BY-ADMIN
current-context: PROVIDED-BY-ADMIN
kind: Config
preferences: {}
users:
- name: PROVIDED-BY-USER
  user:
    auth-provider:
      config:
        client-id: pks_cluster_client
        cluster_client_secret: ""
        id-token: PROVIDED-BY-USER
        idp-issuer-url: <span>https:</span>//PROVIDED-BY-ADMIN:8443/oauth/token
        refresh-token:  PROVIDED-BY-USER
      name: oidc
```

Obtain Cluster Access as a User

To obtain cluster access, the end user must perform the following actions:

1. Run the following command to obtain the `users.user.auth-provider.config.id-token` and `users.user.auth-provider.config.refresh-token`:

   ```
   curl 'https://PKS-API:8443/oauth/token' -k -XPOST -H
   'Accept: application/json' -d "client_id=pks_cluster_client&client_secret=""&grant_type=password
    &username=UAA-USERNAME&response_type=id_token" --data-urlencode password=UAA-PASSWORD
   ```

   Where:

   - `PKS-API` is the FQDN you use to access the PKS API.
   - `UAA-USERNAME` is the Kubernetes end user's UAA username.
   - `UAA-PASSWORD` is the Kubernetes end user's UAA password.

2. Edit the `kubeconfig` by providing the following:

   - `contexts.context.user`
   - `users.name`
   - `users.user.auth-provider.config.id-token`
   - `users.user.auth-provider.config.refresh-token`

3. Save the `kubeconfig` to the `$HOME/.kube/config` directory. After doing so, the Kubernetes end user can connect to the cluster using `kubectl`.

Note: To automate this process, follow the instructions in one of the following Knowledge Base Articles:

- Script to automate generation of the kubeconfig for the kubernetes user
- Powershell script to automate generation of kubeconfig for the kubernetes user

Grant Cluster Access to a Group

Cluster admins can also grant cluster-wide access to an LDAP Group by creating a `ClusterRoleBinding` for that LDAP group. This feature is only available if LDAP is used as your identity provider for UAA.

Note: You must confirm that the group you are referencing in your `ClusterRoleBinding` has been whitelisted in the PKS tile. To do so, review the **External Groups Whitelist** field in the UAA section of the PKS tile.

The process for granting cluster access to an LDAP is similar to the process described in Grant Cluster Access to a User.

The only difference is that when the cluster admin is creating the spec file containing the `RoleBinding` or `ClusterRoleBinding` for a group, the spec file must reflect the following:

```
kind: ROLE-BINDING-TYPE
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: ROLE-OR-CLUSTER-ROLE-BINDING-NAME
  namespace: NAMESPACE
subjects:
- kind: Group
  name: NAME-OF-GROUP
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ROLE-TYPE
  name: ROLE-OR-CLUSTER-ROLE-NAME
  apiGroup: rbac.authorization.k8s.io
```

Where:

- `ROLE-BINDING-TYPE` is the type of role binding you are creating. This must be either `RoleBinding` or `ClusterRoleBinding`.
- `ROLE-OR-CLUSTER-ROLE-BINDING-NAME` is the name of your `RoleBinding` or `ClusterRoleBinding`. This is created by the cluster admin.
- `NAME-OF-GROUP` is the LDAP group name. This name is case sensitive.
- `ROLE-TYPE` is the type of role you are creating. This must be either `Role` or `ClusterRole`.
- `ROLE-OR-CLUSTER-ROLE-NAME` is the name of your `Role` or `ClusterRole`. This is created by the cluster admin.

Please send any feedback you have to pks-feedback@pivotal.io.

Managing PKS Deployments with BOSH

Page last updated:

This topic describes how to manage your Pivotal Container Service (PKS) deployment using BOSH.

Set a BOSH Environment Alias

To set a BOSH alias for your PKS deployment environment, follow the steps below:

1. Gather credential and IP address information for your BOSH Director and SSH into the Ops Manager VM. See Advanced Troubleshooting with the BOSH CLI for more information.

2. To create a BOSH alias for your PKS environment, run the following command:

   ```
   bosh alias-env ENVIRONMENT \
   -e BOSH-DIRECTOR-IP \
   --ca-cert /var/tempest/workspaces/default/root_ca_certificate
   ```

   Where:

   - `ENVIRONMENT` is an alias of your choice. For example, `pks`.
   - `BOSH-DIRECTOR-IP` is the BOSH Director IP address you located in the first step. For example, `10.0.0.3`.

   For example:

   ```
   $ bosh alias-env pks -e 10.0.0.3 \
   --ca-cert /var/tempest/workspaces/default/root_ca_certificate
   ```

3. To log in to the BOSH Director using the alias you set, run the following command:

   ```
   bosh -e ENVIRONMENT login
   ```

   For example:

   ```
   $ bosh -e pks login
   ```

SSH into the PKS API VM

To SSH into the PKS API VM using BOSH, follow the steps below:

1. Gather credential and IP address information for your BOSH Director, SSH into the Ops Manager VM, and use BOSH CLI to log in to the BOSH Director from the Ops Manager VM. For more information, see Advanced Troubleshooting with the BOSH CLI.

2. To identify your PKS deployment's name, run the following command:

   ```
   bosh -e ENVIRONMENT deployments
   ```

   Where `ENVIRONMENT` is the BOSH environment alias you set in Set a BOSH Environment Alias.

   For example:

   ```
   $ bosh -e pks deployments
   ```

   Your PKS deployment name begins with `pivotal-container-service` and includes a BOSH-generated hash.

3. To identify your PKS VM's name, run the following command:

   ```
   bosh -e ENVIRONMENT -d DEPLOYMENT vms
   ```

   Where:

- `ENVIRONMENT` is the BOSH environment alias.
- `DEPLOYMENT` is your PKS deployment name.

For example:

```
$ bosh -e pks -d pivotal-container-service/a1b2c333d444e5f66a77 vms
```

Your PKS VM name begins with `pivotal-container-service` and includes a BOSH-generated hash.

Note: The PKS VM hash value is different from the hash in your PKS deployment name.

4. To SSH into the PKS VM, run the following command:

```
bosh -e ENVIRONMENT -d DEPLOYMENT ssh PKS-VM
```

Where:

- `ENVIRONMENT` is the BOSH environment alias.
- `DEPLOYMENT` is your PKS deployment name.
- `PKS-VM` is your PKS VM name.

For example:

```
$ bosh -e pks \
-d pivotal-container-service/a1b2c333d444e5f66a77 \
ssh pivotal-container-service/000a1111-222b-3333-4cc5-de66f7a8899b
```

Please send any feedback you have to pks-feedback@pivotal.io.

PersistentVolume Storage Options on vSphere

Page last updated:

This topic describes options for configuring Pivotal Container Service (PKS) on vSphere to support stateful apps using PersistentVolumes (PVs).

Note: This topic assumes that you have strong familiarity with PVs and workloads in Kubernetes.

For procedural information about configuring PVs, see Configuring and Using PersistentVolumes.

Considerations for Running Stateful Apps in Kubernetes

There are several factors to consider when running stateful apps in Kubernetes:

- **Pods are ephemeral by nature**. Data that needs to be persisted must be accessible on restart and rescheduling of a pod.
- **When a pod is rescheduled, it may be on a different host**. Storage must be available on the new host for the pod to start gracefully.
- **The app should not manage the volume and data**. The underlying infrastructure should handle the complexity of unmounting and mounting.
- **Certain apps have a strong sense of identity**. When a container with a certain ID uses a disk, the disk becomes tied to that container. If a pod with a certain ID gets rescheduled, the disk associated with that ID must be reattached to the new pod instance.

Persistent Volume Provisioning Support in Kubernetes

Kubernetes provides two ways to provision persistent storage for stateful applications:

- **Static provisioning**: A Kubernetes administrator creates the Virtual Machine Disk (VMDK) and PVs. Developers issue PersistentVolumeClaims (PVCs) on the pre-defined PVs.
- **Dynamic provisioning**: Developers issue PVCs against a StorageClass object. The provisioning of the persistent storage depends on the infrastructure. With PKS on vSphere, the vSphere Cloud Provider (VCP) automatically provisions the VMDK and PVs.

For more information about PVs in Kubernetes, refer to the Kubernetes documentation.

PVs can be used with two types of Kubernetes workloads:

- Deployments
- StatefulSets

vSphere Support for Static and Dynamic PVs

With PKS on vSphere, you can choose one of two storage options to support stateful apps:

- vSAN datastores
- Network File Share (NFS) or VMFS over Internet Small Computer Systems Interface (iSCSI), or fiber channel (FC) datastores

Refer to the vSAN documentation and the VMFS documentation for more information about these storage options.

Note: This topic assumes that you have strong familiarity vSAN and VMFS storage technologies on the vSphere platform.

In PKS, an availability zone (AZ) corresponds to a vSphere cluster and a resource pool within that cluster. A resource pool is a vSphere construct that is not linked to a particular ESXi host. Resource pools can be used in testing environments to enable a single vSphere cluster to support multiple AZs. As a recommended practice, deploy multiple AZs across different vSphere clusters to afford best availability in production.

The vSAN datastore boundary is delimited by the vSphere cluster. All ESXi hosts in the same vSphere cluster belong to the same vSAN datastore. ESXi hosts in a different vSphere cluster belong to a different vSAN datastore. Each vSphere cluster has its own vSAN datastore.

The table below summarizes PKS support for PVs in Kubernetes when deployed on vSphere:

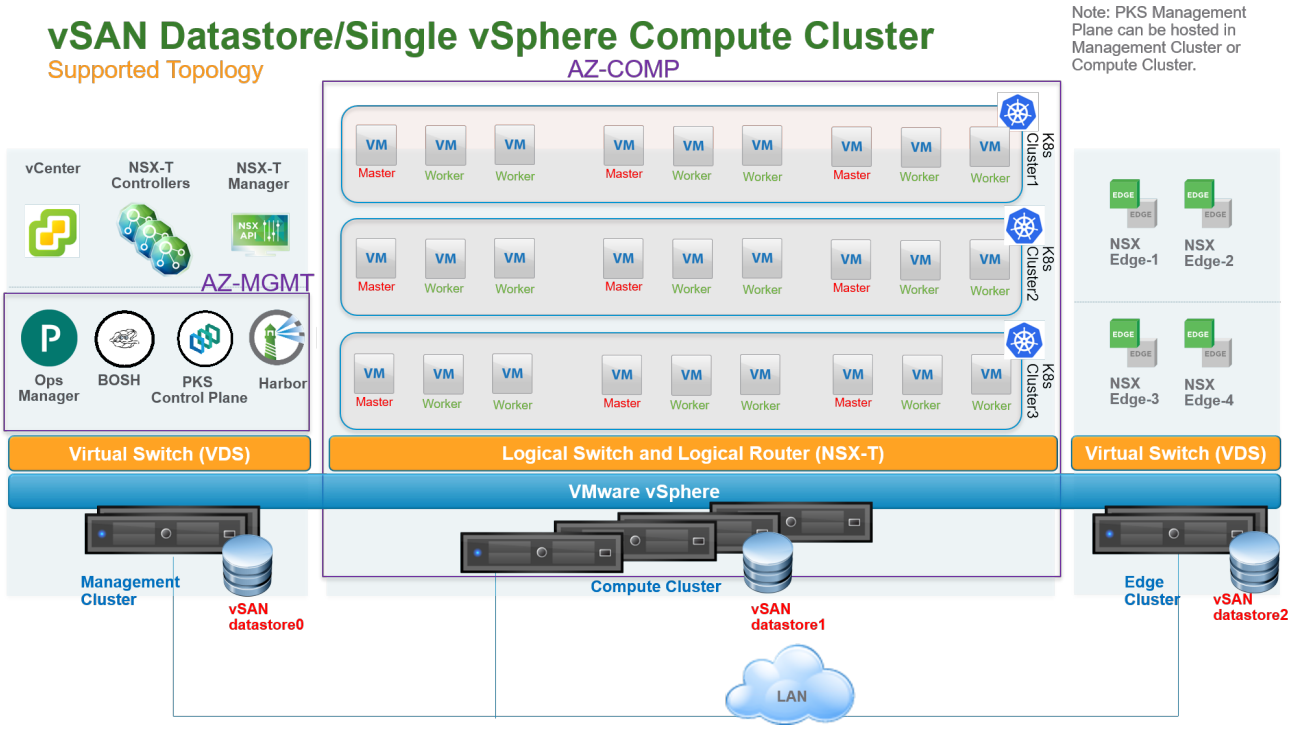
Storage Mechanism	vSAN datastore	File system datastore (VMFS/NFS over iSCSI/FC)

• Single vSphere compute cluster with single datastore • Single AZ and resource pool	Both static and dynamic PV provisioning are supported.	Both static and dynamic PV provisioning are supported.
• Multiple vSphere compute clusters each with local datastore • Multiple AZs each using separate resource pool	Neither static nor dynamic PV provisioning are supported.	Neither static nor dynamic PV provisioning are supported.
• Multiple vSphere compute clusters with shared datastore • Multiple AZs using a shared resource pool	vSAN does not support sharing datastores across vSphere clusters. Can be accomplished by providing vSphere clusters with access to additional shared storage such as VMFS/NFS over iSCSI/FC.	Both static and dynamic PV provisioning are supported.

Note: This information assumes that the underlying vSphere infrastructure is a single locality environment where all vSphere compute clusters are closed in terms of distance from one to the others. It does not apply to multi-site or vSAN stretched cluster configurations.

Single vSphere Compute Cluster with vSAN Datastore

The following diagram illustrates a vSphere environment with a single compute cluster and a local vSAN datastore. This topology is also supported for environments with a single AZ or multiple AZs using multiple resource pools under the same vSphere cluster. For this topology, PKS supports both static and dynamic PV provisioning. Dynamic PV provisioning is recommended.



In this topology, a single vSphere compute cluster hosts all Kubernetes clusters. vSAN is enabled on the compute cluster which exposes a single unique vSAN datastore. In the above diagram, this datastore is labeled **vSAN datastore1**.

You can configure a single computer cluster in the following ways:

- If you use a single PKS foundation, create an AZ that is mapped directly to the single cluster.
- If you use multiple PKS foundations, create an AZ that is mapped to this single cluster and a Resource Pool.

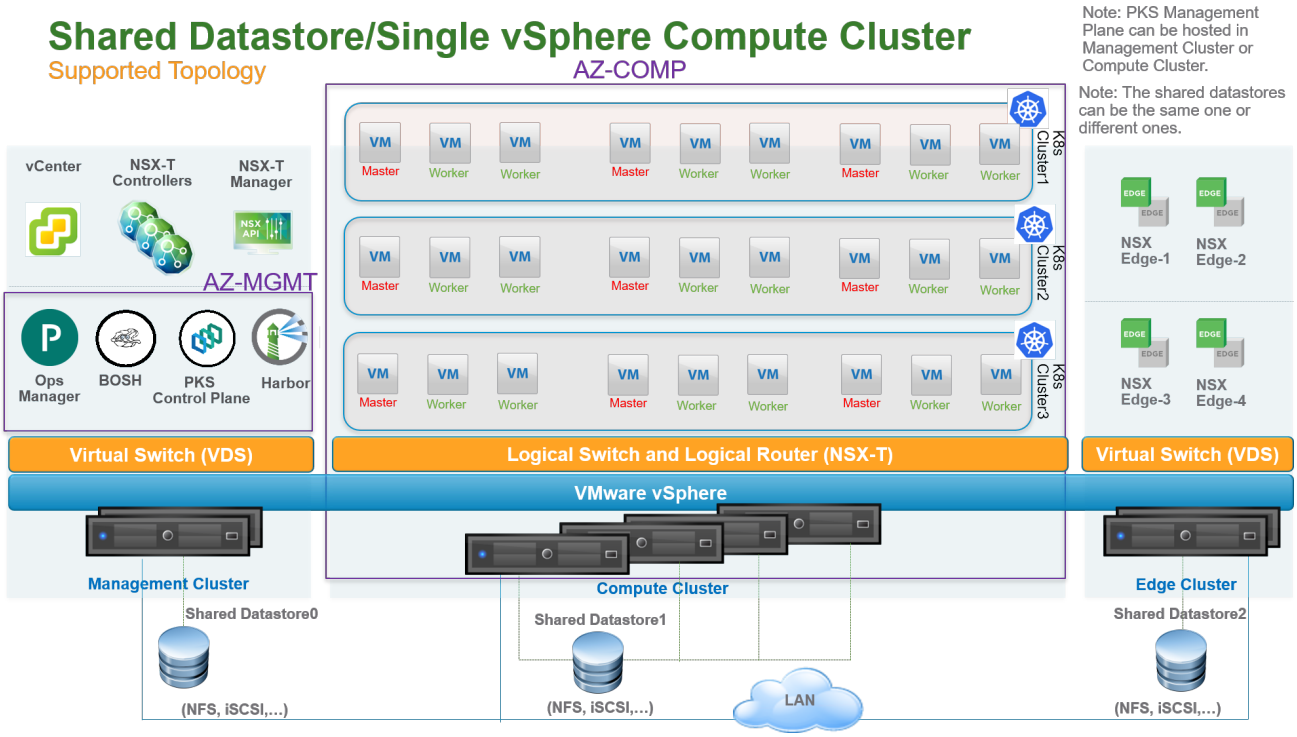
With this topology, you can create multiple vSAN datastores on the same compute cluster using different disk groups on each ESXi host. PVs, backed by respective VMDK files, can be dispatched across the datastores to mitigate the impact of datastore failure. For StatefulSets, all PVs used by different instances of the replica land in the same datastore.

This topology has the following failover scenarios:

- **Disks on ESXi hosts are down**: If the failure is within the limit of the vSAN `failure to tolerate` value, there is no impact on PVs.
- **ESXi hosts are down**: If the failure is within the limit of the vSAN `failure to tolerate` value, there is no impact on PVs.
- **Datastore is down**: PVs on the down datastore are unreachable.

Single vSphere Compute Cluster with File System Datastore

The following diagram illustrates a vSphere environment with a single vSphere compute cluster and a shared datastore using NFS or VMFS over iSCSI, or FC. For this topology, PKS supports both static and dynamic PV provisioning. Dynamic PV provisioning is recommended.



In this topology, a single vSphere compute cluster hosts all Kubernetes clusters. The shared datastore is used with the compute cluster. In the above diagram, this datastore is labeled **Shared Datastore1**.

One or more AZs can be instantiated on top of the compute cluster. With this configuration, one or more AZs are mapped to vSphere resource pools. The AZ is not bound to a failure domain because its resource pool is not linked to a particular ESXi host.

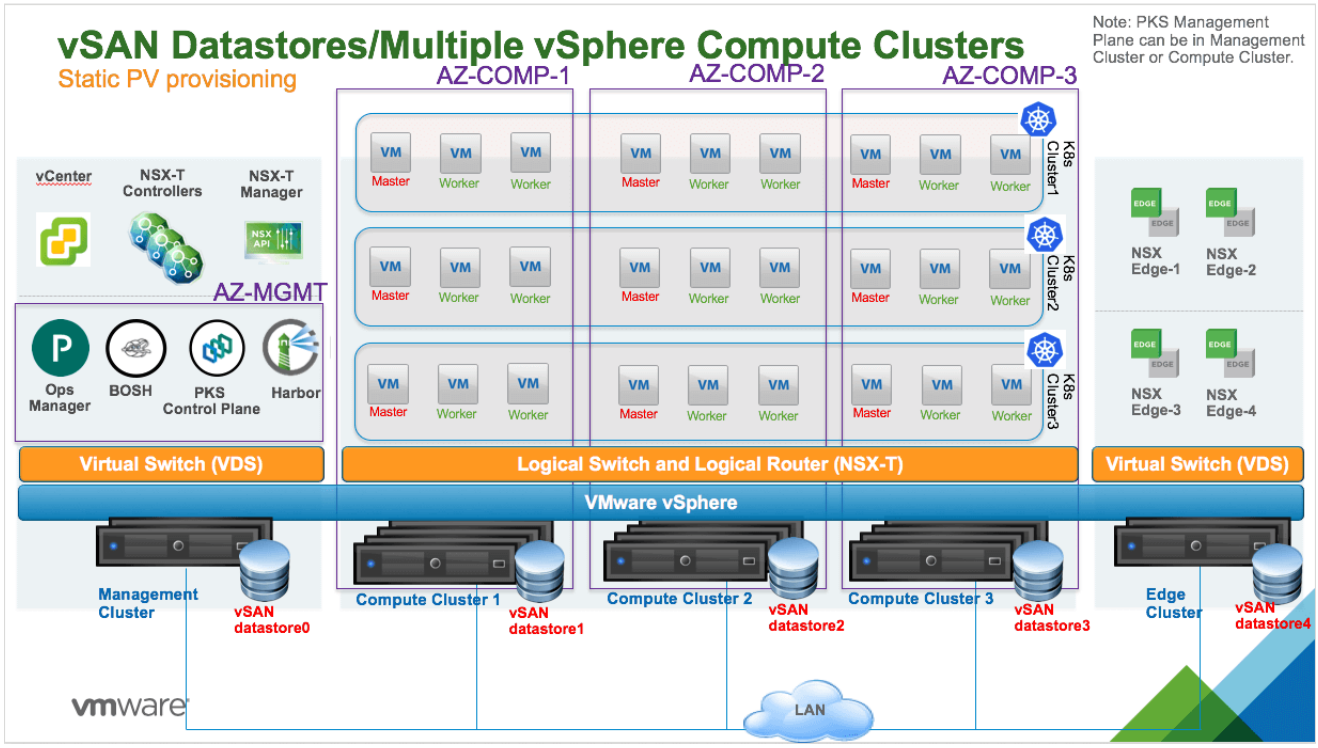
With this topology, you can create multiple shared datastores connected to the same compute cluster. PVs, backed by respective VMDK files, can be dispatched across the datastores to mitigate the impact of datastore failure. For StatefulSets, all PVs used by different instances of the replica land in the same datastore.

This topology has the following failover scenarios:

- **ESXi hosts are down**: No impact on PVs.
- **Datastore is down**: PVs on the down datastore are unreachable.

Multiple vSphere Compute Clusters Each with vSAN Datastore

The following diagram illustrates a vSphere environment with multiple vSphere compute clusters with vSAN datastores that are local to each compute cluster.

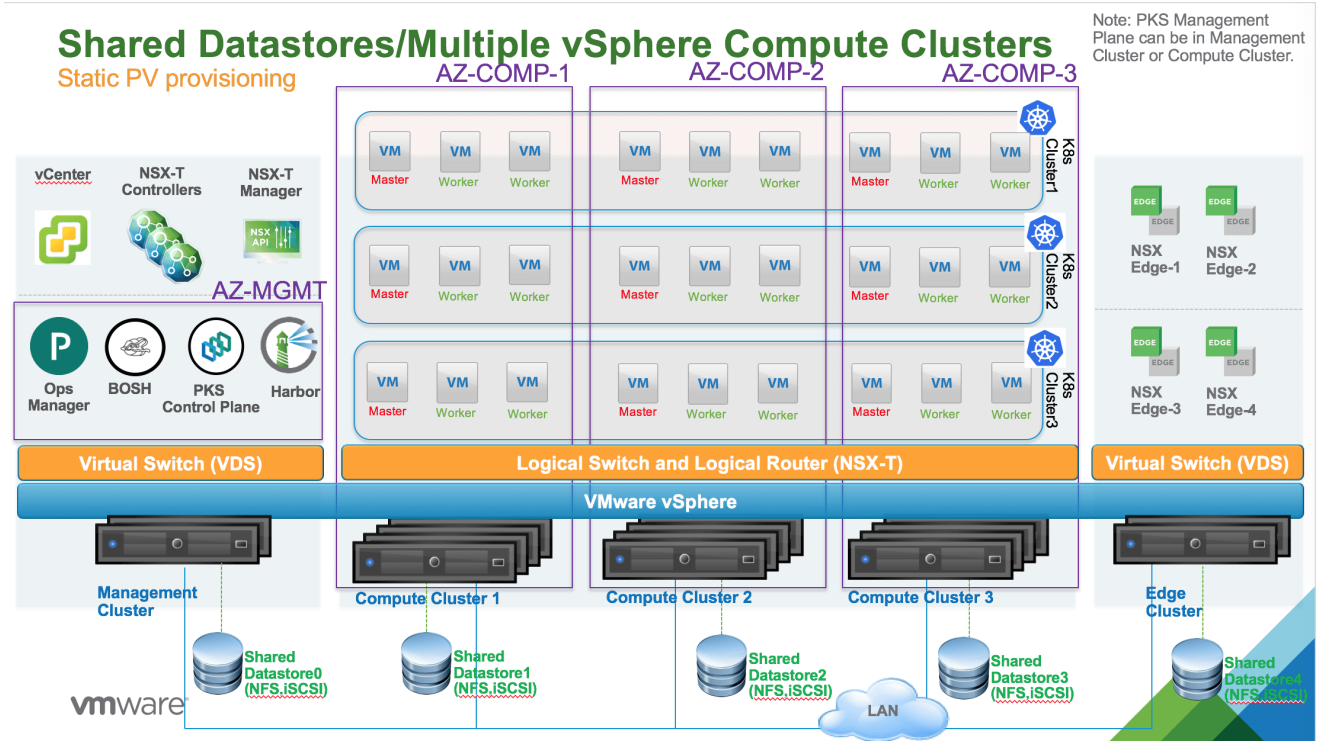


In this topology, vSAN is enabled on each compute cluster. There is one local vSAN datastore per compute cluster. For example, in the above diagram, vSAN datastore1 is provisioned for Compute Cluster 1 and vSAN datastore2 is provisioned for Compute Cluster 2.

One or more AZs can be instantiated. Each AZ is mapped to a vSphere compute cluster. The AZ is bound to a failure domain which is typically the physical rack where the compute cluster is hosted.

Multiple vSphere Compute Clusters Each with File System Datastore

The following diagram illustrates a vSphere environment with multiple vSphere compute clusters with NFS or VMFS over iSCSI, or FC shared datastores.



In this topology, multiple vSphere compute clusters are used to host all Kubernetes clusters. A unique shared datastore is used per vSphere compute cluster. For example, in the above diagram, Shared Datastore1 is connected to Compute Cluster 1 and Shared Datastore2 is connected to Compute

Cluster 2.

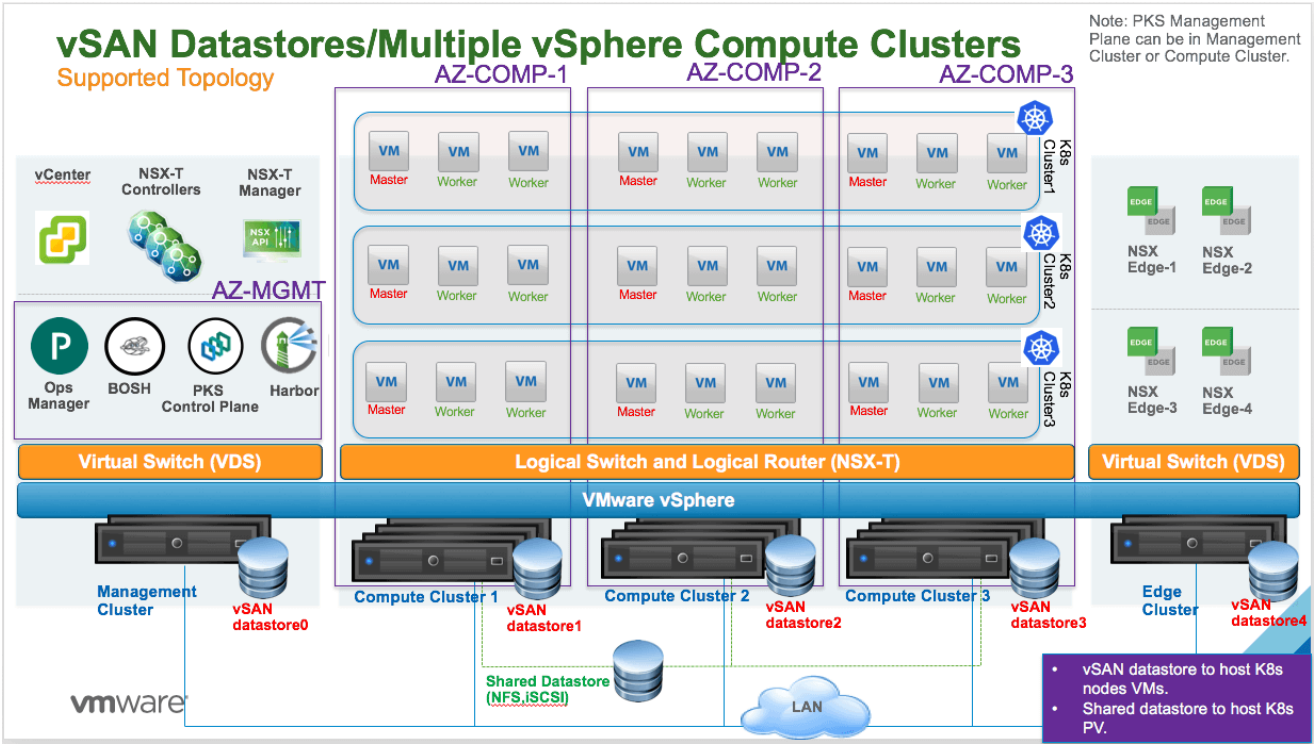
One or more AZs can be instantiated. Each AZ is mapped to a vSphere compute cluster. The AZ is bound to a failure domain which is typically the physical rack where the compute cluster is hosted.

Multiple vSphere Compute Clusters with Local vSAN and Shared File System Datastore

With this topology, each vSAN datastore is only visible from each vSphere compute cluster. It is not possible to have a vSAN datastore shared across all vSphere compute clusters.

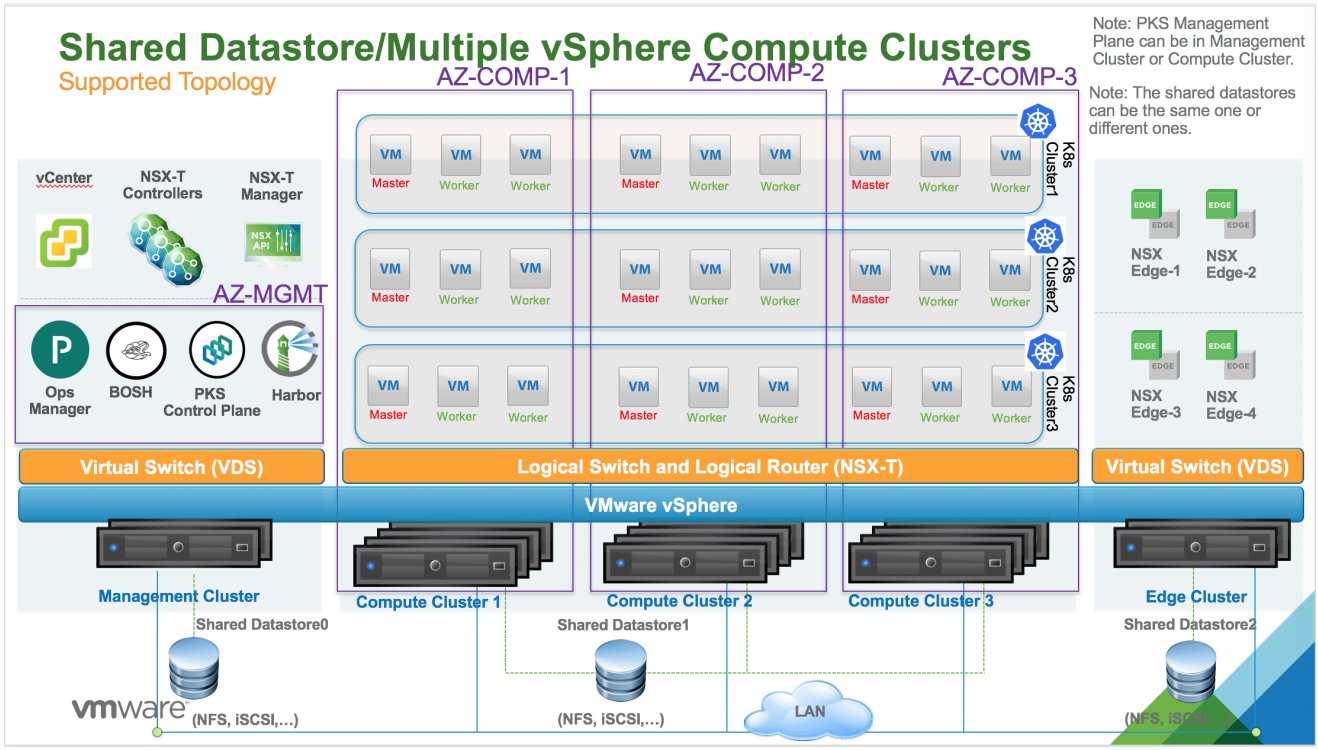
You can insert a shared NFS, iSCSI (VMFS), or FC (VMFS) datastore across all vSAN-based vSphere compute clusters to support both static and dynamic PV provisioning.

Refer to the following diagram:



Multiple vSphere Compute Clusters with Shared File System Datastore

The following diagram illustrates a vSphere environment with multiple compute clusters with VMFS over NFS, iSCSI, or FC datastores shared across all vSphere compute clusters. For this topology, PKS supports both static and dynamic PV provisioning. Dynamic PV provisioning is recommended.



In this topology, multiple vSphere compute clusters are used to host all Kubernetes clusters. A unique shared datastore that uses NFS, or VMFS over iSCSI/FC is used across all compute clusters. In the above diagram, this datastore is labeled **Shared Datastore1**.

One or more AZs can be instantiated. Each AZ is mapped to a compute cluster. The AZ is bound to a failure domain which is typically the physical rack where the compute cluster is hosted.

You can have multiple shared datastores connected across all the vSphere compute clusters. PVs, backed by respective VMDK files, can then be dispatched across those datastores to mitigate the impact of datastore failure. For StatefulSets, all PVs used by different instances of the replica land in the same datastore.

This topology has the following failover scenarios:

- **ESXi hosts are down**: No impact on PVs.
- **One shared datastore is down**: PVs on the down datastore are unreachable.

Please send any feedback you have to pks-feedback@pivotal.io.

Adding Custom Workloads

Page last updated:

This topic describes how to add custom workloads to Pivotal Container Service (PKS) clusters.

Custom workloads define what a cluster includes out of the box. For example, you can use custom workloads to configure metrics or logging.

Create YAML Configuration

Create a YAML configuration for your custom workloads. Consult the following example from the Kubernetes documentation:

```
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2 # tells deployment to run 2 pods matching the template
  template: # create pods using pod definition in this template
    metadata:
      # unlike pod-nginx.yaml, the name is not included in the meta data as a unique name is
      # generated from the deployment name
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

Apply Custom Workloads

To apply custom Kubernetes workloads to every cluster created on a plan, enter your YAML configuration in the **(Optional) Add-ons - Use with caution** field in the pane for configuring a plan in the PKS tile.

For more information, see the *Plans* section of the *Installing PKS* topic for your IaaS. For example, Plans in *Installing PKS on vSphere*.

Please send any feedback you have to pks-feedback@pivotal.io.

Configuring Ingress Routing

Page last updated:

This topic provides resources for configuring an ingress controller on Pivotal Container Service (PKS).

For information about configuring an ingress controller using NSX-T, see Configuring Ingress Resources and Load Balancer Services.

Overview

In Kubernetes, an ingress is an API object that manages external access to the services in a cluster. You can use ingress rules to provide HTTP or HTTPS routes to services within the cluster instead of creating a load balancer. For more information, see Ingress in the Kubernetes documentation.

The cluster must have an ingress controller running. You define ingress resource configuration in the manifest of your Kubernetes deployment, and then use wildcard DNS entries to route traffic to the exposed ingress resource.

To configure an ingress controller, you must do the following:

1. Deploy a Kubernetes Ingress Controller
2. Configure DNS
3. (Optional) Configure TLS
4. Deploy an App to the Cluster

Prerequisites

Before you configure an ingress controller, you must have the following:

- A PKS-deployed cluster with its own load balancer. See Creating Clusters.
- A wildcard DNS record that points to the cluster load balancer.

Deploy a Kubernetes Ingress Controller

You can deploy an ingress controller of your choice to your Kubernetes cluster. For a list of ingress controllers that Kubernetes supports, see Ingress Controllers in the Kubernetes documentation.

> **Note**: For information about configuring an ingress controller using NGINX on Amazon Web Services (AWS), Azure, or Google Cloud Platform (GCP), see How to set up an Ingress Controller for a PKS cluster in the Pivotal Knowledge Base.

To deploy an open source ingress controller to a PKS cluster, do the following:

1. Set the kubectl context for the cluster where you want to deploy the ingress controller by running the following command:

   ```
   pks get-credentials CLUSTER-NAME
   ```

   Where `CLUSTER-NAME` is the name of your PKS-deployed Kubernetes cluster.

2. Verify that KubeDNS is enabled for your cluster by running the following command:

   ```
   kubectl cluster-info
   ```

   If KubeDNS is enabled, the output lists the URL for the KubeDNS service in the cluster. For example:

```
$ kubectl cluster-info
Kubernetes master is running at https://104.197.5.247
elasticsearch-logging is running at https://104.197.5.247/api/v1/namespaces/kube-system/services/elasticsearch-logging/proxy
kibana-logging is running at https://104.197.5.247/api/v1/namespaces/kube-system/services/kibana-logging/proxy
kube-dns is running at https://104.197.5.247/api/v1/namespaces/kube-system/services/kube-dns/proxy
grafana is running at https://104.197.5.247/api/v1/namespaces/kube-system/services/monitoring-grafana/proxy
heapster is running at https://104.197.5.247/api/v1/namespaces/kube-system/services/monitoring-heapster/proxy
```

If KubeDNS is not enabled, do the following:

a. Navigate to Ops Manager and click the **BOSH Director** tile.
b. Click the **Director Config** pane.
c. Select the **Enable Post Deploy Scripts** checkbox.
d. Click **Review Pending Changes**, and then **Apply Changes**.
e. Delete the cluster, and then re-create the cluster.

3. Follow the installation instructions for the Kubernetes ingress controller you choose to deploy. For example, see the installation guide in the Istio documentation.

Configure DNS

After you deploy an ingress controller to your cluster, locate the HTTP port number that the ingress rules expose. Configure DNS to point to the exposed port on your Kubernetes worker node VMs.

To configure DNS for your cluster, do the following:

1. Run `kubectl get services` in the namespace where you deployed the ingress controller. For example, if you deployed Istio, run the following command:

```
kubectl  --namespace=istio-system get services
```

In the output of this command, locate the exposed HTTP port.

For example:

```
$ kubectl --namespace=istio-system get services
NAME           TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)
istio-ingress  LoadBalancer   10.100.200.200  <pending>       80:30822/TCP,443:31441/TCP
```

In the example above, the exposed HTTP port is 30822.

2. List the IP addresses for the Kubernetes worker node VMs by running the following command:

```
kubectl -o jsonpath='{.items[*].status.addresses[0].address}' get nodes
```

3. Configure your load balancer to point to the Kubernetes worker node VMs, using the IP addresses you located in the previous step and the exposed port number you located in the first step.

(Optional) Configure TLS

Enable Transport Layer Security (TLS) for the domain you configured for the cluster.

To configure TLS, do the following:

1. (Optional) Run the following command to generate a self-signed certificate:

```
openssl req -x509 \
  -nodes -newkey rsa:4096 \
  -keyout KEY-PATH.pem \
  -out CERT-PATH.pem \
  -days 365 \
  -subj "/CN=*.PKS.EXAMPLE.COM"
```

Where:

- `KEY-PATH.pem` is the file path for the key you are generating.

- `CERT-PATH.pem` is the file path for the certificate you are generating.
- `*.PKS.EXAMPLE.COM` is the wildcard domain you configured in Configure DNS.

2. Upload your TLS certificate and key to your ingress controller namespace by running the following command:

```
kubectl -n INGRESS-NAMESPACE create secret tls INGRESS-CERT \
--key `KEY-PATH.pem` --cert CERT-PATH.pem
```

Where:

- `INGRESS-CERT` is a name you provide for the Kubernetes secret that contains your TLS certificate and key pair.
- `KEY-PATH.pem` is the file path for your TLS key.
- `CERT-PATH.pem` is the file path for your TLS certificate.

For example:

```
$ kubectl -n istio-system create secret tls istio-ingress-certs \
--key /tmp/tls.key --cert /tmp/tls.crt
```

Deploy an App to the Cluster

When your cluster has an ingress controller running and DNS configured, you can deploy an app to the cluster that uses the ingress rules.

To deploy an app that uses ingress rules, do the following:

1. Deploy your app manifest by running the following command:

```
kubectl create -f YOUR-APP.yml
```

Where `YOUR-APP.yml` is the file path for your app manifest.

2. In the app manifest for your ingress controller, change the value of the `host:` property to match the wildcard domain you configured in Configure DNS above.

3. Deploy your ingress controller app manifest by running the following command:

```
kubectl create -f YOUR-APP.yml
```

Where `INGRESS-CONTROLLER.yml` is the file path for your ingress controller app manifest.

4. Navigate to the fully qualified domain name (FQDN) you defined in your app manifest and confirm that you can access your app workload.

5. (Optional) If you configured TLS, do the following:

   a. Add the following to your ingress controller manifest to enable TLS:

```
spec:
  tls:
    - secretName: INGRESS-CERT
  rules:
  - host: INGRESS.PKS.EXAMPLE.COM
```

   Where:

   - `INGRESS-CERT` is the name of the Kubernetes secret that contains your TLS certificate and key pair.
   - `INGRESS.PKS.EXAMPLE.COM` is the domain you defined for your app in the app manifest.

   b. Redeploy the ingress controller manifest to update the ingress service by running the following command:

```
kubectl replace -f INGRESS-CONTROLLER.yml
```

   Where `INGRESS-CONTROLLER.yml` is the file path for your ingress controller app manifest.

   c. Navigate to the FQDN you defined in your app manifest and confirm that you can access your app workload.

Deleting PKS

This topic explains how to delete the Pivotal Container Service (PKS) tile.

Delete the PKS Tile

To delete the PKS tile, perform the following steps:

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the trash can icon on the PKS tile.
3. Click **Confirm**.
4. Click **Review Pending Changes**.
5. (Optional) By default, deleting the PKS tile also deletes all the clusters created by PKS. To preserve the clusters, click **Errands** and deselect the **Delete all clusters** errand.
6. Click **Apply Changes**.

Shutting Down and Starting Up PKS

Page last updated:

This topic lists and describes the shutdown and startup sequence for Pivotal Container Service (PKS) including PKS-provisioned Kubernetes cluster nodes, PKS control plane components, and vSphere hosts.

Shutdown Sequence and Tasks

To perform a graceful shutdown of all Kubernetes, PKS, and infrastructure components, complete the following tasks in sequence.

Step 1: Shut Down Customer Apps

Shut down all customer apps running on PKS-provisioned Kubernetes clusters.

Note: This task is optional. Perform it after considering the types of apps you have deployed. For example, stateful, stateless, or legacy apps.

Step 2: Shut Down Kubernetes Clusters

Shut down all PKS-provisioned Kubernetes clusters following the procedure defined in the How to shutdown and startup a Multi Master PKS cluster knowledge base article.

For each Kubernetes cluster that you intend to shut down, do the following:

1. Using the BOSH CLI, retrieve the BOSH deployment name of your PKS clusters by running the following command.

   ```
   bosh deployments
   ```

   Kubernetes cluster deployment names begin with `service-instance_` and include a unique BOSH-generated hash.

2. Using the BOSH CLI, stop the Kubernetes worker nodes by running the following command.

   ```
   bosh -d service-instance_CLUSTER-UUID stop worker
   ```

   Where `CLUSTER-UUID` is the BOSH deployment name of your PKS cluster. For example:

   ```
   $ bosh -d service-instance_aa1234567bc8de9f0a1c stop worker
   ```

   Note: When you use the BOSH `stop` command, all processes on the Kubernetes node are stopped. BOSH marks them stopped so that when the VM is powered back on, the processes do not start automatically.

3. Using the BOSH CLI, stop the Kubernetes master nodes by running the following command.

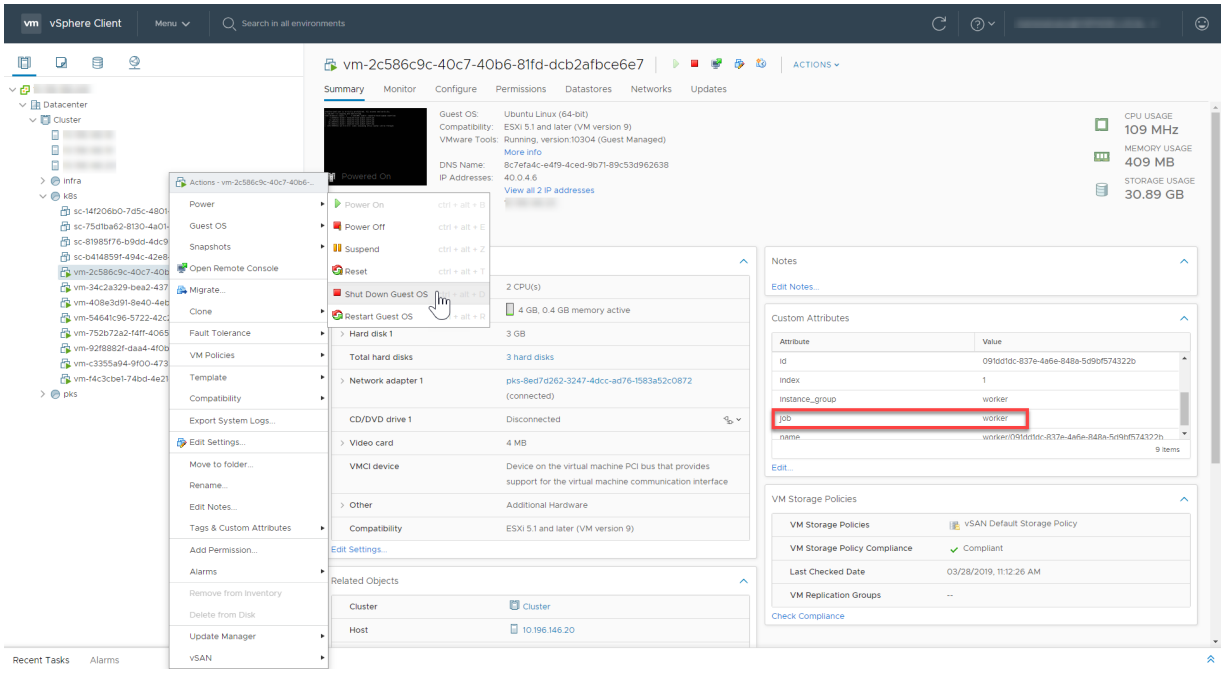
   ```
   bosh -d service-instance_CLUSTER-UUID stop master
   ```

   Where `CLUSTER-UUID` is the BOSH deployment name of your PKS cluster. For example:

   ```
   $ bosh -d service-instance_aa1234567bc8de9f0a1c stop master
   ```

4. Using vCenter, shut down all Kubernetes node VMs. To do this, perform the following steps:

   a. Verify the node type by checking the “job” name in the the **Custom Attributes** pane.
   b. Perform a graceful shutdown by right-clicking the target VM and selecting **Power** > **Shut Down Guest OS**.



View a larger version of this image.

Step 3: Shut Down PKS Control Plane

To shut down the PKS control plane VM, do the following:

1. Using the BOSH CLI, retrieve the BOSH deployment ID of your PKS deployment by running the following command.

    ```
    bosh deployments
    ```

    PKS deployment names begin with `pivotal-container-service` and include a unique BOSH-generated hash.

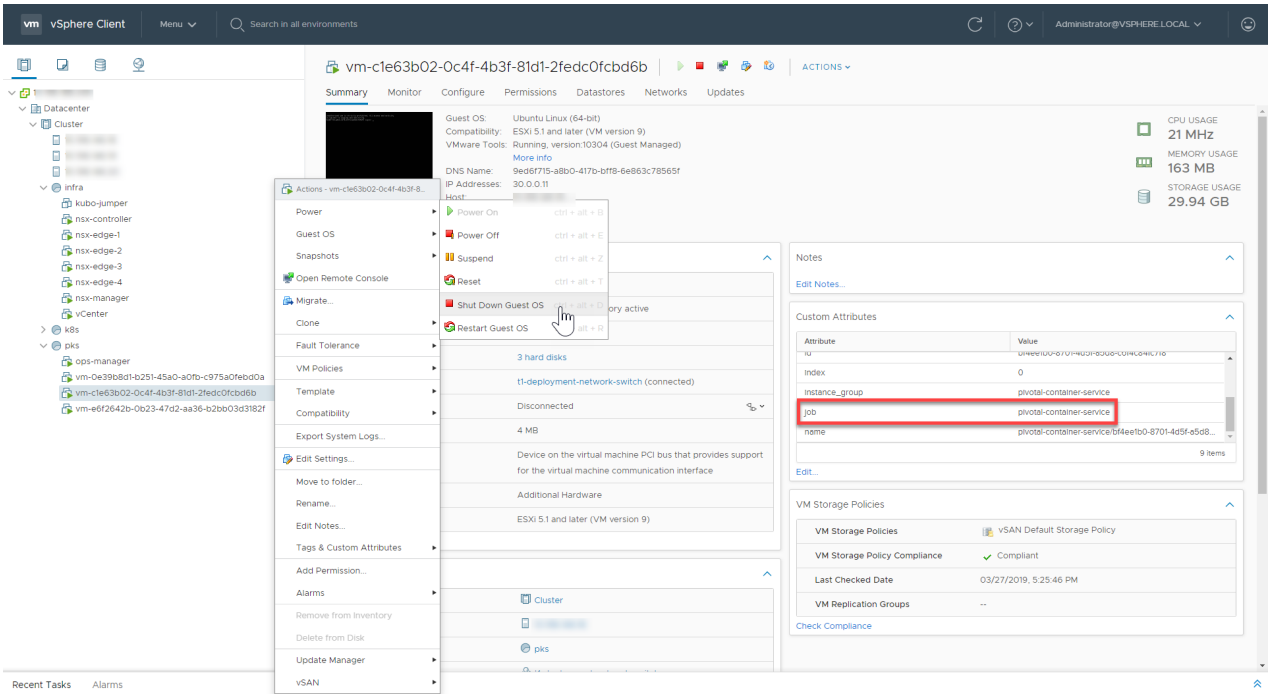
2. Using the BOSH CLI, stop the PKS control plane VM by running the following command.

    ```
    bosh -d pivotal-container-service-DEPLOYMENT-ID stop
    ```

    Where `DEPLOYMENT-ID` is the BOSH-generated ID of your PKS deployment. For example:

    ```
    $ bosh -d pivotal-container-service-1bf7b02738056cdc37e6 stop
    ```

3. Using vCenter, locate and gracefully shut down the PKS control plane VM.



View a larger version of this image.

Step 4: Shut Down VMware Harbor Registry

To shut down the Harbor Registry VM, do the following:

1. Using the BOSH CLI, retrieve the BOSH deployment ID of your Harbor Registry deployment by running the following command.

    ```
    bosh deployments
    ```

    Harbor Registry deployment names begin with `harbor-container-registry` and include a unique BOSH-generated hash.

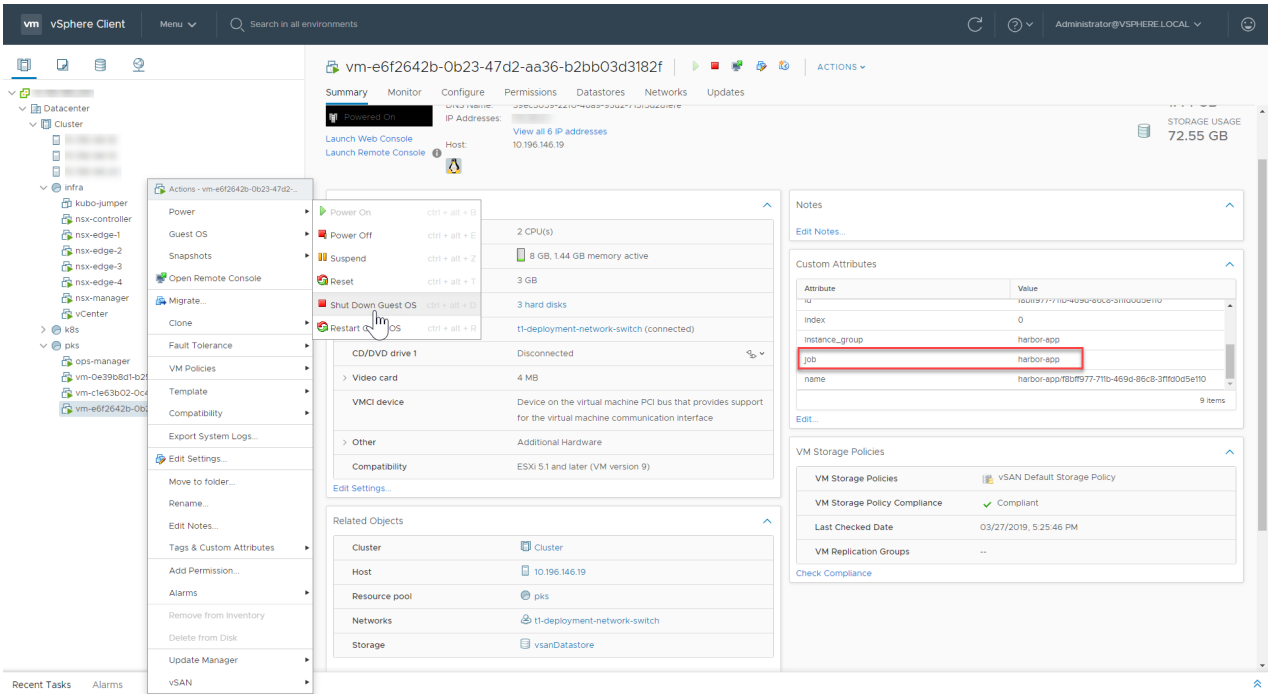
2. Using the BOSH CLI, stop the Harbor Registry VM by running the following command.

    ```
    bosh -d harbor-container-registry-DEPLOYMENT-ID stop
    ```

    Where `DEPLOYMENT-ID` is the BOSH-generated ID of your Harbor Registry deployment. For example:

    ```
    $ bosh -d harbor-container-registry-b4023f6857207b237399 stop
    ```

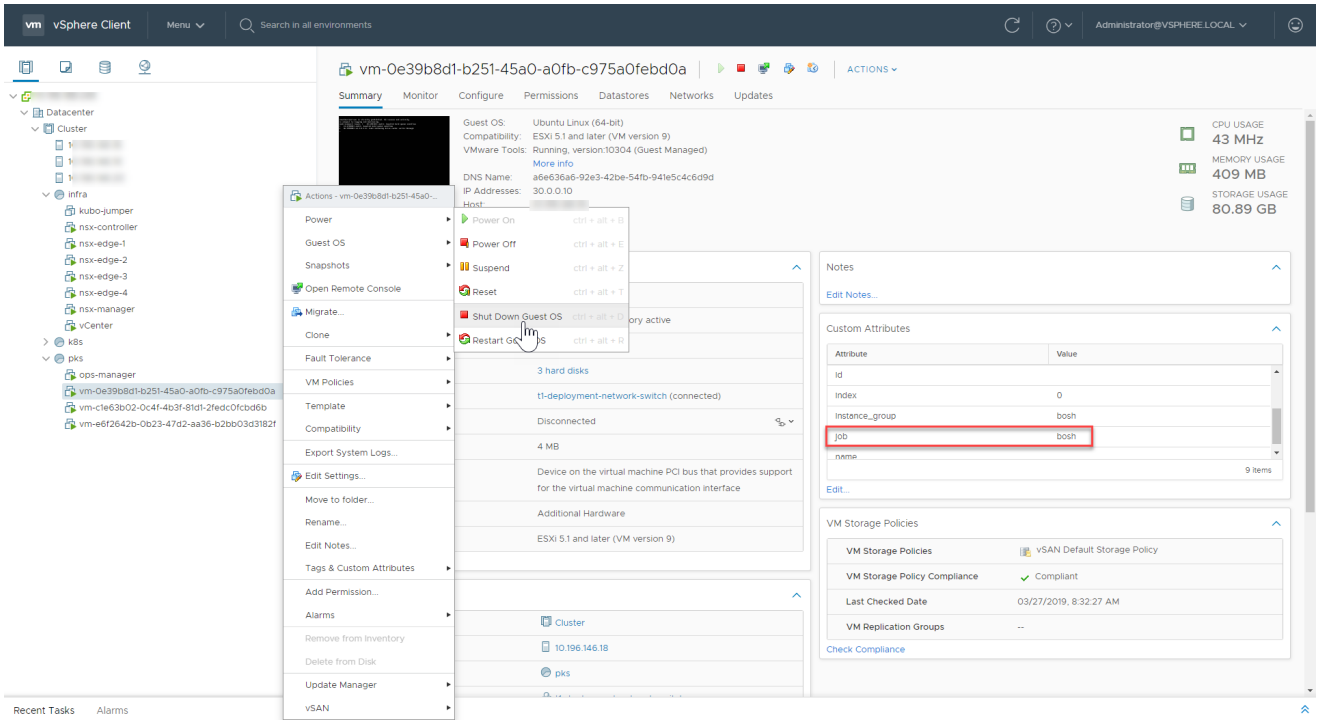
3. Using vCenter, locate and gracefully shut down the Harbor Registry VM.



View a larger version of this image.

Step 5: Shut Down BOSH Director

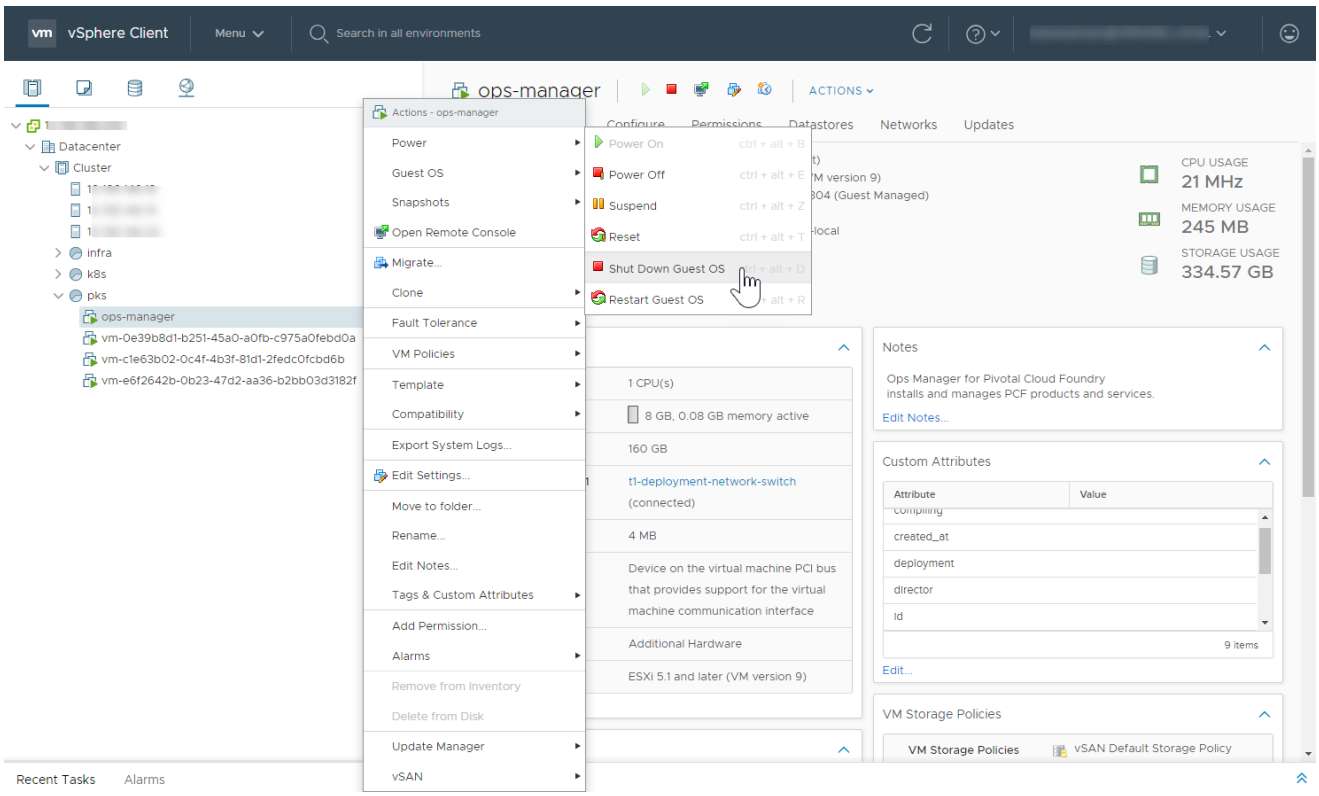
Using vCenter, locate and gracefully shut down the BOSH Director VM.



View a larger version of this image.

Step 6: Shut Down Ops Manager

Using vCenter, locate and gracefully shut down the Ops Manager VM.

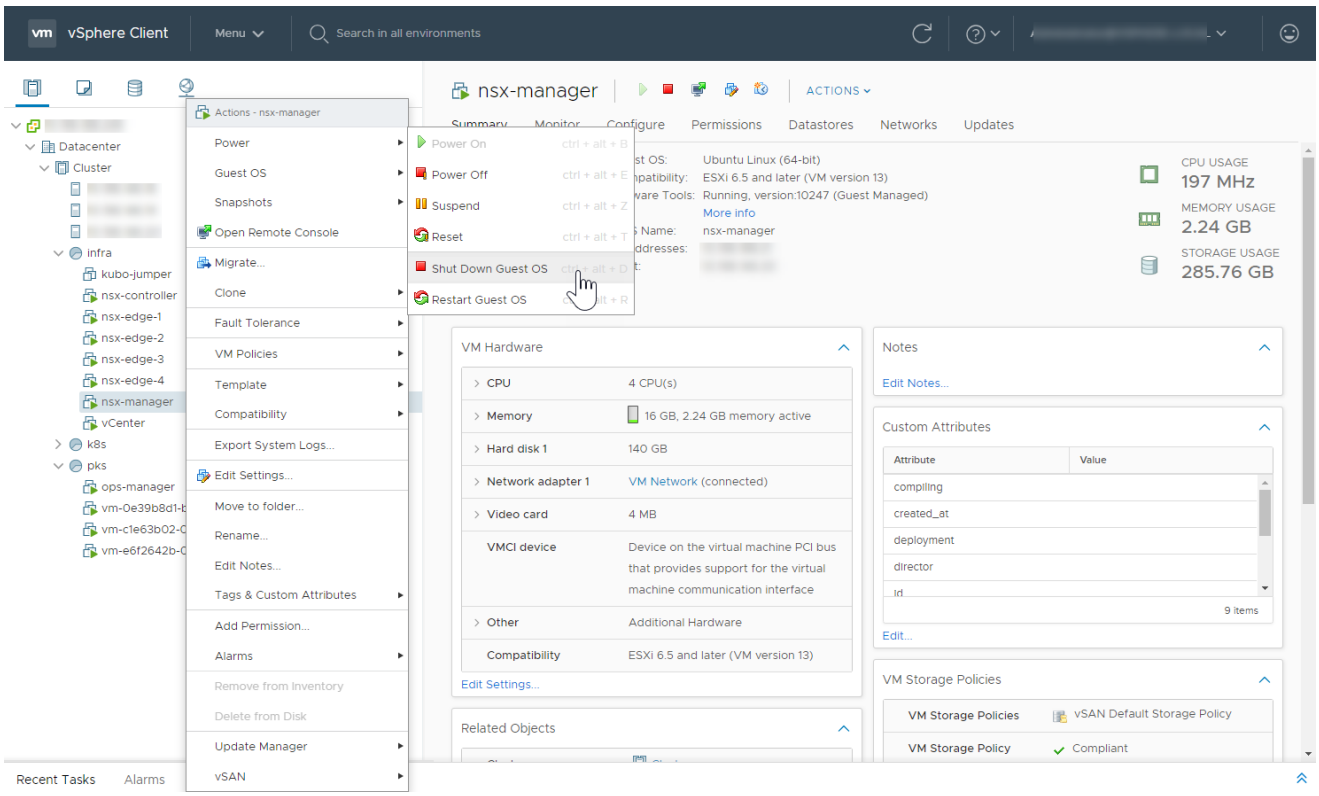


View a larger version of this image.

Step 7: Shut Down NSX-T Components

Using vCenter, gracefully shut down all NSX-T VMs in the following order:

1. NSX-T Manager
2. NSX-T Controllers
3. NSX-T Edge Nodes



View a larger version of this image.

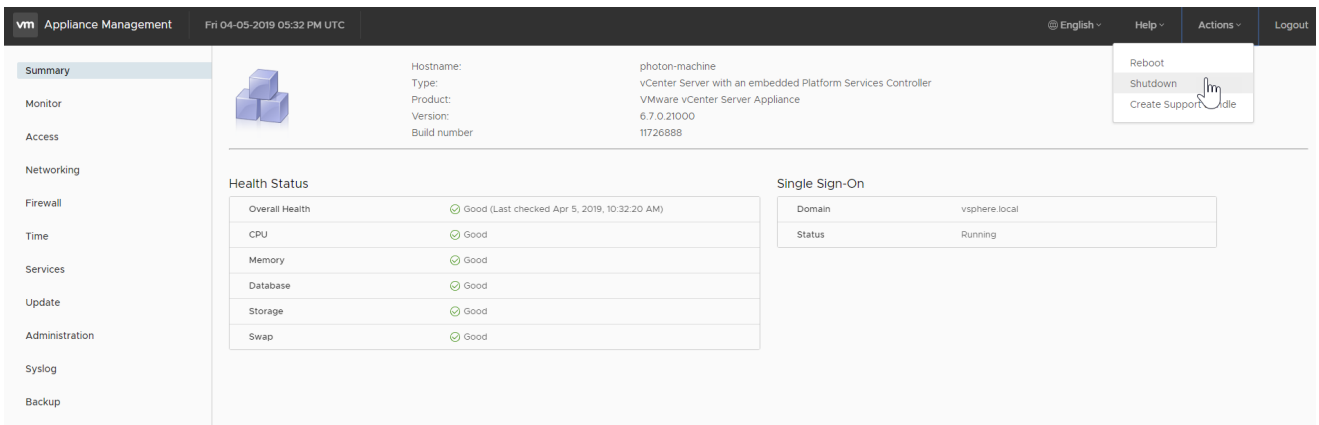
Step 8: Shut Down vCenter Server

To shut down the vCenter Server VM, do the following:

1. Navigate to the vCenter Appliance Management Interface at `https://YOUR-VCENTER-HOSTNAME-OR-IP-ADDRESS>:5480`, where `YOUR-VCENTER-HOSTNAME-OR-IP-ADDRESS` is the hostname or IP address that you use to connect to vCenter through the vSphere Web Client.
2. Log in as root.
3. Select **Actions** > **Shutdown** from the menu and confirm the operation.

For more information about how to shut down the vCenter Server VM, see Reboot or Shut Down the vCenter Server Appliance in the vSphere documentation and the How to stop, start, or restart vCenter Server 6.x services KB article.

Note: After you shut down this vCenter VM, the vSphere Web Client will not be available.

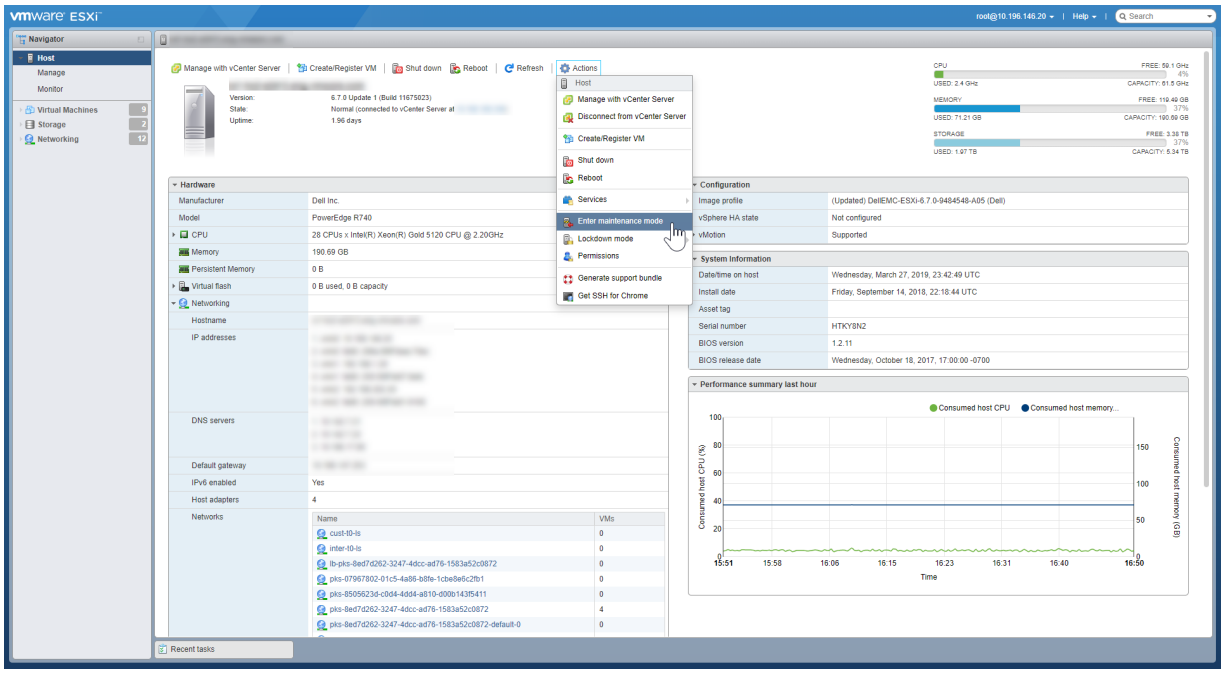


View a larger version of this image

Step 9: Shut Down ESXi Hosts

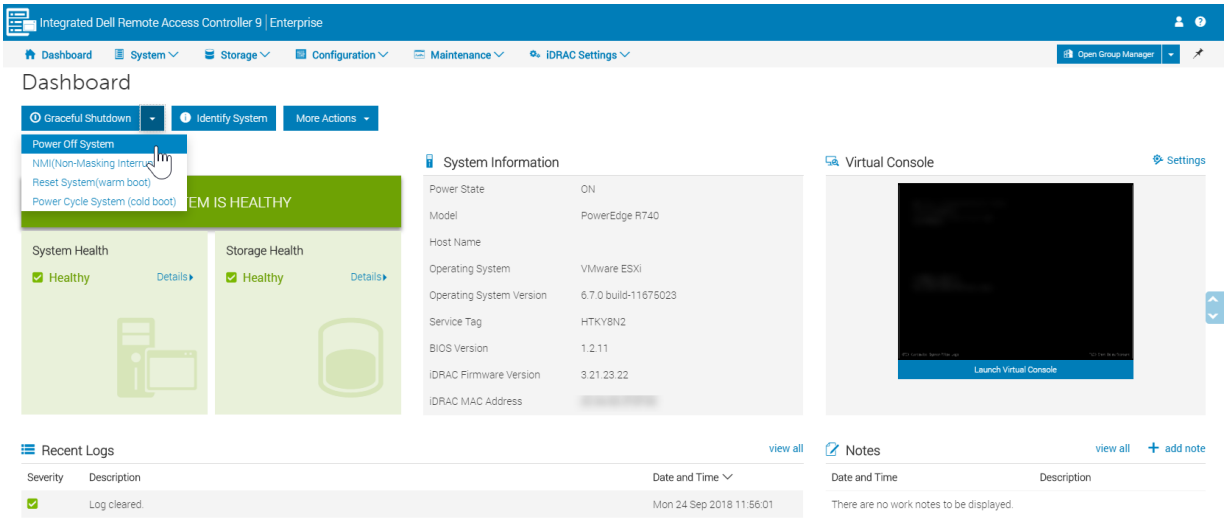
To shut down each ESXi host in the vSphere cluster, do the following:

1. Put the ESXi host into maintenance mode by doing the following:
    a. Using a browser, navigate to the HTTPS IP address of the ESXi host, for example: https://10.196.146.20/ .
    b. Log in using vSphere administrative credentials.
    c. Put the ESXi host in maintenance mode by selecting **Actions** > **Enter maintenance mode**.



View a larger version of this image.

2. Power off the ESXi host. To do this, you have two options:
    - Use the EXSi web interface and select **Actions** > **Shut down**.
    - Use the remote management console for the host, such as Dell IDRAC or HP iLO.



View a larger version of this image.

Startup Sequence and Tasks

To restart all Kubernetes, PKS, and infrastructure components, complete the following tasks in the sequence presented.

Step 1: Start ESXi Hosts

To start the ESXi hosts, do the following:

1. Using the remote management console, such as Dell IDRAC or HP iLO, power on each ESXi host.
2. Connect to the web interface of each ESXi host and exit maintenance mode.

Step 2: Start vCenter

Connect to the web interface of the ESXi server that hosts the vCenter VM. Select the vCenter VM, and click **Power On**.

Step 3: Start NSX-T Components

To start the NSX-T components, perform the following steps:

1. Log into vCenter using the vSphere Client.
2. Power on the following VMs in the following order:
    a. NSX-T Manager
    b. NSX-T Controllers
    c. NSX-T Edge Nodes

Step 4: Start Ops Manager

Using vCenter, power on the Ops Manager VM.

Step 5: Start the BOSH Director

Using vCenter, power on the BOSH Director VM.

Note: BOSH is aware that all the VMs under its control were stopped. BOSH does not attempt to resurrect any VMs, which is the desired behavior.

It may take approximately 90 minutes for BOSH to start properly.

To speed up the BOSH startup process:

1. Obtain the BOSH Director VM Credentials from Ops Manager. For information about doing this, see Retrieving Credentials from Your Deployment in the Pivotal documentation.
2. SSH to the BOSH Director VM.
3. On the BOSH Director VM, run the following commands:

```
sudo -i
monit summary
```

4. If you see messages such as `Process uaa Connection failed` and `Process credhub not monitored`, then run the following command:

```
monit restart uaa
```

5. After a few minutes, run the following command again:

```
monit summary
```

You should see that the `uaa` and `credhub` processes are now running. At this point, the BOSH Director should be fully up and running.

Step 6: Start the PKS Control Plane

To start the PKS control plane, do the following:

1. Using vCenter, power on the PKS control plane VM.
2. Using the BOSH CLI, start the PKS process on the VM by running the following command.

   ```
   bosh -d pivotal-container-service-DEPLOYMENT-ID start
   ```

   Where `DEPLOYMENT-ID` is the BOSH-generated ID of the PKS deployment. For example:

   ```
   $ bosh -d pivotal-container-service-1bf7b02738056cdc37e6 start
   ```

   Note: Because you stopped the PKS process using BOSH, you must restart it using BOSH.

Step 7: Start Harbor Registry

To start Harbor Registry, do the following:

1. Using vCenter, power on the Harbor VM.
2. Using the BOSH CLI, start the Harbor process on the VM by running the following command:

   ```
   bosh -d harbor-container-registry-DEPLOYMENT-ID start
   ```

   Where `DEPLOYMENT-ID` is the BOSH-generated ID of your Harbor Registry deployment. For example:

   ```
   $ bosh -d harbor-container-registry-b4023f6857207b237399 start
   ```

Step 8: Start the Kubernetes Clusters

For each Kubernetes cluster that you intend to start up, start the Kubernetes nodes in the following order:

1. Using the BOSH CLL, run `ssh` to access the first PKS master node and start etcd.
2. Using the BOSH CLI, start the next PKS master node.
3. Using the BOSH CLI, start all remaining PKS master nodes including the master where you started etcd.
4. Using the BOSH CLI, start all PKS worker nodes.

For exact Kubernetes node startup instructions, refer to the How to shutdown and startup a Multi Master PKS cluster knowledge base article.

Step 9: Start Customer Apps

Start all apps running on the PKS-provisioned Kubernetes clusters.

Please send any feedback you have to pks-feedback@pivotal.io.

Managing Clusters

Page last updated:

This section describes how to manage Pivotal Container Service (PKS) clusters.

See the following topics:

- Creating Clusters
- Using Network Profiles (NSX-T Only)
- Retrieving Cluster Credentials and Configuration
- Viewing Cluster Lists
- Viewing Cluster Details
- Viewing Cluster Plans
- Scaling Existing Clusters
- Deleting Clusters

Please send any feedback you have to pks-feedback@pivotal.io.

Creating Clusters

Page last updated:

This topic describes how to create a Kubernetes cluster with Pivotal Container Service (PKS) using the PKS Command Line Interface (PKS CLI).

Configure Cluster Access

Cluster access configuration differs by the type of PKS deployment.

vSphere with NSX-T

PKS deploys a load balancer automatically when clusters are created. The load balancer is configured automatically when workloads are being deployed on these Kubernetes clusters. For more information, see Load Balancers in PKS Deployments with NSX-T.

Note: For a complete list of the objects that PKS creates by default when you create a Kubernetes cluster on vSphere with NSX-T, see vSphere with NSX-T Cluster Objects.

GCP, AWS, Azure, or vSphere without NSX-T

When you create a Kubernetes cluster, you must configure external access to the cluster by creating an external TCP or HTTPS load balancer. This load balancer allows you to run PKS CLI commands on the cluster from your local workstation. For more information, see Load Balancers in PKS Deployments without NSX-T.

You can configure any load balancer of your choice. If you use GCP, AWS, Azure, or vSphere without NSX-T, you can create a load balancer using your cloud provider console.

For more information about configuring a PKS cluster load balancer, see the following:

- Creating and Configuring a GCP Load Balancer for PKS Clusters
- Creating and Configuring an AWS Load Balancer for PKS Clusters
- Creating and Configuring an Azure Load Balancer for PKS Clusters

Create the PKS cluster load balancer before you create the cluster. Use the load balancer IP address as the external hostname, and then point the load balancer to the IP address of the master virtual machine (VM) after cluster creation. If the cluster has multiple master nodes, you must configure the load balancer to point to all master VMs for the cluster.

If you are creating a cluster in a non-production environment, you can choose to create a cluster without a load balancer. Create a DNS entry that points to the IP address of the cluster's master VM after cluster creation.

To locate the IP addresses and VM IDs of the master VMs, see Identify the Kubernetes Cluster Master VM below.

Create a Kubernetes Cluster

Perform the following steps:

1. Grant cluster access to a new or existing user in UAA. See the Grant PKS Access to a User section of *Managing Users in PKS with UAA* for more information.
2. On the command line, run the following command to log in:

   ```
   pks login -a PKS-API -u USERNAME -k
   ```

   Where:

   - `PKS-API` is the domain name for the PKS API that you entered in **Ops Manager** > **Pivotal Container Service** > **PKS API** > **API Hostname**

(FQDN). For example, `api.pks.example.com`.
- `USERNAME` is your user name.

  See Logging in to PKS for more information about the `pks login` command.

3. To create a cluster run the following command :

```
pks create-cluster CLUSTER-NAME \
--external-hostname HOSTNAME \
--plan PLAN-NAME \
[--num-nodes WORKER-NODES] \
[--network-profile NETWORK-PROFILE-NAME]
```

Where:

- `CLUSTER-NAME` is your unique name for your cluster.

  Note: The `CLUSTER-NAME` must not contain special characters such as `&`. The PKS CLI does not validate the presence of special characters in the `CLUSTER-NAME` string, but cluster creation fails if one or more special characters are present.

- `HOSTNAME` is your external hostname for your cluster. You can use any fully qualified domain name (FQDN) or IP address you own. For example, `my-cluster.example.com` or `10.0.0.1`. If you created an external load balancer, use its DNS hostname.
- `PLAN-NAME` is the plan for your cluster. Run `pks plans` to list your available plans.
- (Optional) `WORKER-NODES` is the number of worker nodes for the cluster.
- (Optional) (NSX-T only) `NETWORK-PROFILE-NAME` is the network profile to use for the cluster. See Using Network Profiles (NSX-T Only) for more information.

For example:

```
$ pks create-cluster my-cluster \
--external-hostname my-cluster.example.com \
--plan large --num-nodes 3
```

Note: It can take up to 30 minutes to create a cluster.

For high availability, create clusters with a minimum of three worker nodes, or two per AZ if you intend to use PersistentVolumes (PVs). For example, if you deploy across three AZs, you should have six worker nodes. For more information about PVs, see PersistentVolumes in *Maintaining Workload Uptime*. Provisioning a minimum of three worker nodes, or two nodes per AZ is also recommended for stateless workloads.

The maximum value you can specify is configured in the **Plans** pane of the Pivotal Container Service tile. If you do not specify a number of worker nodes, the cluster is deployed with the default number, which is also configured in the **Plans** pane. For more information, see the *Installing PKS* topic for your IaaS, such as Installing PKS on vSphere.

4. To track cluster creation, run the following command:

```
pks cluster CLUSTER-NAME
```

Where `CLUSTER-NAME` is the unique name for your cluster.

For example:

```
$ pks cluster my-cluster
Name:                  my-cluster
Plan Name:             large
UUID:                  01a234bc-d56e-7f89-01a2-3b4cde5f6789
Last Action:           CREATE
Last Action State:     succeeded
Last Action Description:  Instance provisioning completed
Kubernetes Master Host:   my-cluster.example.com
Kubernetes Master Port:   8443
Worker Instances:      3
Kubernetes Master IP(s):  192.168.20.7
```

5. If the **Last Action State** value is `error`, troubleshoot by performing the following procedure:

   a. Log in to the BOSH Director.
   b. Run the following command:

```
bosh tasks
```

For more information, see Advanced Troubleshooting with the BOSH CLI.

6. Depending on your deployment:
   - For **vSphere with NSX-T**, choose one of the following:
     - Specify the hostname or FQDN and register the FQDN with the IP provided by PKS after cluster deployment. You can do this using `resolv.conf` or via DNS registration.
     - Specify a temporary placeholder value for FQDN, then replace the FQDN in the `kubeconfig` with the IP address assigned to the load balancer dedicated to the cluster.

       To retrieve the IP address to access the Kubernetes API and UI services, use the `pks cluster CLUSTER-NAME` command.
   - For **vSphere without NSX-T**, **AWS**, and **Azure**, configure external access to the cluster's master nodes using either DNS records or an external load balancer. Use the output from the `pks cluster` command to locate the master node IP addresses and ports.
   - For **GCP**, use the output from the `pks cluster` command to locate the master node IP addresses and ports, and then continue to Configure Load Balancer Backend in *Configuring a GCP Load Balancer for PKS Clusters*.

   Note: For clusters with multiple master node VMs, health checks on port 8443 are recommended.

7. To access your cluster, run the following command:

   ```
   pks get-credentials CLUSTER-NAME
   ```

   Where `CLUSTER-NAME` is the unique name for your cluster.

   For example:

   ```
   $ pks get-credentials pks-example-cluster


   Fetching credentials for cluster pks-example-cluster.
   Context set for cluster pks-example-cluster.


   You can now switch between clusters by using:
   $kubectl config use-context &ltcluster-name&gt
   ```

   The `pks get-credentials` command creates a local `kubeconfig` that allows you to manage the cluster. For more information about the `pks get-credentials` command, see Retrieving Cluster Credentials and Configuration.

8. To confirm you can access your cluster using the Kubernetes CLI, run the following command:

   ```
   kubectl cluster-info
   ```

   See Managing PKS for information about checking cluster health and viewing cluster logs.

Identify Kubernetes Cluster Master VMs

Note: This section applies only to PKS deployments on GCP or on vSphere without NSX-T. Skip this section if your PKS deployment is on vSphere with NSX-T, AWS, or Azure. For more information, see Load Balancers in PKS.

To reconfigure the load balancer or DNS record for an existing cluster, you may need to locate VM ID and IP address information for the cluster's master VMs. Use the information you locate in this procedure when configuring your load balancer backend.

To locate the IP addresses and VM IDs for the master VMs of an existing cluster, do the following:

1. On the command line, run the following command to log in:

   ```
   pks login -a PKS-API -u USERNAME -k
   ```

Where:

- `PKS-API` is the domain name for the PKS API that you entered in **Ops Manager** > **Pivotal Container Service** > **PKS API** > **API Hostname (FQDN)**. For example, `api.pks.example.com`.
- `USERNAME` is your user name.

  See Logging in to PKS for more information about the `pks login` command.

2. To locate the cluster ID and master node IP addresses, run the following command:

   ```
   pks cluster CLUSTER-NAME
   ```

   Where `CLUSTER-NAME` is the unique name for your cluster.

   From the output of this command, record the following items:

   - **UUID**: This value is your cluster ID.
   - **Kubernetes Master IP(s)**: This value lists the IP addresses of all master nodes in the cluster.

3. Gather credential and IP address information for your BOSH Director.

4. To log in to the BOSH Director, perform the following:

   a. SSH into the Ops Manager VM.
   b. Log in to the BOSH Director by using the BOSH CLI from the Ops Manager VM.

   For information on how to complete these steps, see Advanced Troubleshooting with the BOSH CLI.

5. To identify the name of your cluster deployment, run the following command:

   ```
   bosh -e pks deployments
   ```

   Your cluster deployment name begins with `service-instance` and includes the UUID you located in a previous step.

6. To identify the master VM IDs by listing the VMs in your cluster, run the following command:

   ```
   bosh -e pks -d CLUSTER-SI-ID vms
   ```

   Where `CLUSTER-SI-ID` is your cluster service instance ID which begins with `service-instance` and includes the `UUID` you previously located.

   For example:

   ```
   $ bosh -e pks -d service-instance-aa1234567bc8de9f0a1c vms
   ```

   Your master VM IDs are displayed in the **VM CID** column.

7. Use the master VM IDs and other information you gathered in this procedure to configure your load balancer backend. For example, if you use GCP, use the master VM IDs retrieved during the previous step in Reconfiguring a GCP Load Balancer.

Next Steps

If your PKS deployment is on AWS, you must tag your subnets with your new cluster's unique identifier before adding the subnets to the PKS workload load balancer. After you complete the Create a Kubernetes Cluster procedure, follow the instructions in AWS Prerequisites.

Using Network Profiles (NSX-T Only)

Page last updated:

This topic describes how to use network profiles for Kubernetes clusters provisioned with Pivotal Container Service (PKS) on vSphere with NSX-T integration. Network profiles let you customize NSX-T configuration parameters.

Assign a Network Profile to a Cluster

You can assign a network profile to a Kubernetes cluster at the time of cluster creation. To assign a network profile to a Kubernetes cluster, you must do the following:

1. Define a network profile configuration in a JSON file. For instructions on how to define network profile configurations, see Defining Network Profiles.
2. Create a network profile using the JSON file. For instructions on how to create network profiles, see Create a Network Profile.
3. Create a Kubernetes cluster with the network profile. For instructions on how to create a Kubernetes cluster with a network profile, see Create a Cluster with a Network Profile.

Note: Only PKS cluster administrators can create and delete network profiles. Cluster managers can list existing network profiles and assign them to clusters.

Create a Cluster with a Network Profile

To create a PKS-provisioned Kubernetes cluster with a network profile, run the following command:

```
pks create-cluster CLUSTER-NAME --external-hostname HOSTNAME --plan PLAN-NAME --network-profile NETWORK-PROFILE-NAME
```

Where:

- `CLUSTER-NAME` is a unique name for your cluster.
- `HOSTNAME` is your external hostname used for accessing the Kubernetes API.
- `PLAN-NAME` is the name of the PKS plan you want to use for your cluster.
- `NETWORK-PROFILE-NAME` is the name of the network profile you want to use for your cluster.

Manage Network Profiles

This section describes how to create, list, and delete network profiles.

Create a Network Profile

After you define your network profile configuration as described in Defining Network Profiles, run the following command:

```
pks create-network-profile PATH-TO-YOUR-NETWORK-PROFILE-CONFIGURATION
```

Where `PATH-TO-YOUR-NETWORK-PROFILE-CONFIGURATION` is the path to the JSON file you created when defining the network profile.

For example:

```
$ pks create-network-profile np-routable-pods.json

Network profile small-routable-pod successfully created
```

Only cluster administrators, `pks.clusters.admin`, can create network profiles. If a cluster manager, `pks.clusters.manage`, attempts to create a network profile,

the following error occurs:

```
You do not have enough privileges to perform this action. Please contact the PKS administrator.
```

List Network Profiles

To list your network profiles, run the following command:

```
pks network-profiles
```

For example:

```
$ pks network-profiles

  Name              Description
  lb-profile-medium   Network profile for medium size NSX-T load balancer
  small-routable-pod  Network profile with small load balancer and two routable pod networks
```

Delete a Network Profile

To delete a network profile, run the following command:

```
pks delete-network-profile NETWORK-PROFILE-NAME
```

Where `NETWORK-PROFILE-NAME` is the name of the network profile you want to delete.

Note: You cannot delete a network profile that is in use.

Only cluster administrators, `pks.clusters.admin`, can delete network profiles. If a cluster manager, `pks.clusters.manage`, attempts to delete a network profile, the following error occurs:

```
You do not have enough privileges to perform this action. Please contact the PKS administrator.
```

Please send any feedback you have to pks-feedback@pivotal.io.

Retrieving Cluster Credentials and Configuration

This topic describes how to use the `pks get-credentials` command in Pivotal Container Service (PKS) using the PKS Command Line Interface (PKS CLI).

The `pks get-credentials` command performs the following actions:

- Fetch the cluster's kubeconfig
- Add the cluster's kubeconfig to the existing kubeconfig
- Create a new kubeconfig, if none exists
- Switch the context to the `CLUSTER-NAME` provided

When you run `pks get-credentials CLUSTER-NAME`, PKS sets the context to the cluster you provide as the `CLUSTER-NAME`. PKS binds your username to the cluster and populates the kubeconfig file on your local workstation with cluster credentials and configuration.

The default path for your kubeconfig is `$HOME/.kube/config`.

If you access multiple clusters, you can choose to use a custom kubeconfig file for each cluster. To save cluster credentials to a custom kubeconfig, use the `KUBECONFIG` environment variable when you run `pks get-credentials`. For example:

```
$ KUBECONFIG=/path/to/my-cluster.config pks get-credentials my-cluster
```

Retrieve Cluster Credentials

Perform the following steps to populate your local kubeconfig with cluster credentials and configuration:

1. On the command line, run the following command to log in:

   ```
   pks login -a PKS-API -u USERNAME -k
   ```

   Where:

   - `PKS-API` is the domain name for the PKS API that you entered in **Ops Manager** > **Pivotal Container Service** > **PKS API** > **API Hostname (FQDN)**. For example, `api.pks.example.com`.
   - `USERNAME` is your user name.

   See Logging in to PKS for more information about the `pks login` command.

2. Run the following command:

   ```
   pks get-credentials CLUSTER-NAME
   ```

   Replace `CLUSTER-NAME` with the unique name for your cluster. For example:

   ```
   $ pks get-credentials my-cluster
   ```

   Note: If you enable OpenID Connect (OIDC) in the PKS tile, PKS requires your password to run the `pks get-credentials CLUSTER-NAME` command. This allows PKS to retrieve valid tokens for the kubeconfig file. You can provide your password at the prompt or as the `PKS_USER_PASSWORD` environment variable. For more information, see the *Configure OpenID Connect* section of Installing PKS for your IaaS.

Run kubectl Commands

After PKS populates your kubeconfig, you can use the Kubernetes Command Line Interface (kubectl) to run commands against your Kubernetes clusters.

See Installing the Kubernetes CLI for information about installing kubectl.

For information about using kubectl, refer to the Kubernetes documentation.

Please send any feedback you have to pks-feedback@pivotal.io.

Viewing Cluster Lists

Follow the steps below to view the list of deployed Kubernetes cluster with the PKS CLI.

1. On the command line, run the following command to log in:

   ```
   pks login -a PKS-API -u USERNAME -k
   ```

   Where:

   - `PKS-API` is the domain name for the PKS API that you entered in **Ops Manager** > **Pivotal Container Service** > **PKS API** > **API Hostname (FQDN)**. For example, `api.pks.example.com`.
   - `USERNAME` is your user name.

   See Logging in to PKS for more information about the `pks login` command.

2. Run the following command to view the list of deployed clusters, including cluster names and status:

   ```
   $ pks clusters
   ```


Viewing Cluster Details

Follow the steps below to view the details of an individual cluster using the PKS CLI.

1. On the command line, run the following command to log in:

   ```
   pks login -a PKS-API -u USERNAME -k
   ```

   Where:

   - `PKS-API` is the domain name for the PKS API that you entered in **Ops Manager** > **Pivotal Container Service** > **PKS API** > **API Hostname (FQDN)**. For example, `api.pks.example.com`.
   - `USERNAME` is your user name.

   See Logging in to PKS for more information about the `pks login` command.

2. Run the following command to view the details of an individual cluster:

   ```
   pks cluster CLUSTER-NAME
   ```

   Replace `CLUSTER-NAME` with the unique name for your cluster. For example:

   ```
   $ pks cluster my-cluster
   ```

Viewing Cluster Plans

Follow the steps below to view information about the available plans for deploying a cluster using the PKS CLI.

1. On the command line, run the following command to log in:

   ```
   pks login -a PKS-API -u USERNAME -k
   ```

   Where:

   - `PKS-API` is the domain name for the PKS API that you entered in **Ops Manager** > **Pivotal Container Service** > **PKS API** > **API Hostname (FQDN)**. For example, `api.pks.example.com`.
   - `USERNAME` is your user name.

   See Logging in to PKS for more information about the `pks login` command.

2. Run the following command to view information about the available plans for deploying a cluster:

   ```
   $ pks plans
   ```

   The response lists details about the available plans, including plan names and descriptions:

   ```
   Name     ID  Description
   default      Default plan for K8s cluster
   ```

Please send any feedback you have to pks-feedback@pivotal.io.

Scaling Existing Clusters

This topic explains how to scale an existing cluster by using the PKS CLI to increase or decrease the number of worker nodes in the cluster.

Follow the steps below to scale an existing cluster using the PKS CLI.

1. On the command line, run the following command to log in:

   ```
   pks login -a PKS-API -u USERNAME -k
   ```

   Where:

   - `PKS-API` is the domain name for the PKS API that you entered in **Ops Manager** > **Pivotal Container Service** > **PKS API** > **API Hostname (FQDN)**. For example, `api.pks.example.com`.
   - `USERNAME` is your user name.

     See Logging in to PKS for more information about the `pks login` command.

2. To view the current number of worker nodes in your cluster, run the following command:

   ```
   pks cluster CLUSTER-NAME
   ```

   Where `CLUSTER-NAME` is the name of your cluster.

3. Run the following command:

   ```
   pks resize CLUSTER-NAME --num-nodes NUMBER-OF-WORKER-NODES
   ```

   Where:

   - `CLUSTER-NAME` is the name of your cluster.
   - `NUMBER-OF-WORKER-NODES` is the number of worker nodes you want to set for the cluster.
     - To scale down your existing cluster, enter a number lower than the current number of worker nodes.
     - To scale up your existing cluster, enter a number higher than the current number of worker nodes. The maximum number of worker nodes you can set is configured in the **Plan** pane of the Pivotal Container Service tile in Pivotal Ops Manager.

   For example:

   ```
   $ pks resize my-cluster --num-nodes 5
   ```

   Note: This command may roll additional virtual machines in the cluster, which can affect workloads if the worker nodes are at capacity.

Deleting Clusters

Page last updated:

This topic describes how to delete a Kubernetes cluster deployed by Pivotal Container Service (PKS). Running the `pks delete-cluster` command automatically deletes all cluster objects.

If you are using PKS with NSX-T, see vSphere with NSX-T Cluster Objects for a list of vSphere and NSX-T objects that will be deleted as part of the cluster deletion process.

Delete Cluster

Follow the steps below to delete a cluster using the PKS CLI.

1. On the command line, run the following command to log in:

   ```
   pks login -a PKS-API -u USERNAME -k
   ```

   Where:

   - `PKS-API` is the domain name for the PKS API that you entered in **Ops Manager** > **Pivotal Container Service** > **PKS API** > **API Hostname (FQDN)**. For example, `api.pks.example.com`.
   - `USERNAME` is your user name.

   See Logging in to PKS for more information about the `pks login` command.

2. Run `pks delete-cluster CLUSTER-NAME` to delete a cluster. Replace `CLUSTER-NAME` with the unique name for your cluster. For example:

   ```
   $ pks delete-cluster my-cluster
   ```

3. Confirm cluster deletion by entering `y`, or cancel cluster deletion by entering `n`.

   For example:

   ```
   Are you sure you want to delete cluster my-cluster? (y/n)
   ```

Verify Cluster Deletion

Follow the steps below to verify cluster deletion using the PKS CLI.

1. To verify cluster deletion, run `pks cluster CLUSTER-NAME`. Replace `CLUSTER-NAME` with the unique name for your cluster.

   For example:

   ```
   $ pks cluster my-cluster
   Name:                     my-cluster
   Plan Name:                small
   UUID:                     106aabc7-5ecb-4c54-a800-a32eef57a593
   Last Action:              DELETE
   Last Action State:        in progress
   Last Action Description:  Instance deletion in progress
   Kubernetes Master Host:   my-cluster.pks.local
   Kubernetes Master Port:   8443
   Worker Nodes:             3
   Kubernetes Master IP(s):  10.196.219.88
   Network Profile Name:
   ```

   While PKS is deleting the cluster, the value for `Last Action Description` is `Instance deletion in progress`.

2. Continue running the `pks cluster CLUSTER-NAME` command to track cluster deletion. The cluster is deleted when the CLI returns `Error: Cluster CLUSTER-NAME not found`.

3. Run `pks clusters`. The cluster you deleted should not appear in the list of PKS clusters.

> **Note**: If the cluster is not deleted, see Cluster Deletion Fails in *Troubleshooting PKS*.

Delete Cluster without Prompt

If you do not want the PKS CLI to prompt you to confirm cluster deletion, use the `--non-interactive` flag.

For example:

```
$ pks delete-cluster my-cluster --non-interactive
```

> **Note**: If you use the `--non-interactive` flag to delete multiple clusters, delete each cluster one by one. Do not create a script that deletes multiple clusters using the `--non-interactive` flag. If you do, the BOSH Director may hang and become unusable until you log in to BOSH and cancel each deletion task.

Please send any feedback you have to pks-feedback@pivotal.io.

Using PKS

Page last updated:

This section describes how to use Pivotal Container Service (PKS).

Note: Because PKS does not currently support the Kubernetes Service Catalog or the GCP Service Broker, binding clusters to Kubernetes services is not supported.

The procedures for using PKS have the following prerequisites:

- You must have an external TCP or HTTPS load balancer configured to forward traffic to the PKS API endpoint. For more information, see the Configure External Load Balancer section of *Installing PKS* for your IaaS.
- You must know the address of your PKS API endpoint and have a UAA-created user account that has been granted PKS cluster access. For more information, see Managing Users in PKS with UAA.

  Note: If your PKS installation is integrated with NSX-T, use the DNAT IP address assigned in the Retrieve the PKS Endpoint section of *Installing PKS on vSphere with NSX-T Integration*.

See the following topics:

- Logging in to PKS
- Accessing Dashboard
- Deploying and Exposing Basic Workloads
- Getting Started with VMware Harbor Registry
- Using Helm with PKS
- Configuring and Using Dynamic PersistentVolumes
- Creating Sink Resources
- Logging Out of PKS

Please send any feedback you have to pks-feedback@pivotal.io.

Logging in to PKS

This topic describes how to log in to Pivotal Container Service (PKS).

Overview

To manage PKS-deployed clusters, you use the PKS Command Line Interface (CLI). When you log in to PKS successfully for the first time, the PKS CLI generates a local `creds.yml` file that contains the API endpoint, refresh token, access token, and CA certificate, if applicable.

By default, `creds.yml` is saved in the `~/.pks` directory on your local system. You can use the `PKS_HOME` environment variable to override this location and store `creds.yml` in any directory on your system.

Prerequisites

Before you can log in to PKS, you must have the following:

- A running PKS environment. See the *Installing PKS* section for your cloud provider.
- The PKS CLI installed on your local system. See Installing the PKS CLI.
- A username and password that has access to the PKS API. See Configuring PKS API Access.

Log in to the PKS CLI

Use the command in this section to log in as an individual user. The login procedure is the same for users created in UAA or users from external LDAP groups.

On the command line, run the following command in your terminal to log in to the PKS CLI:

```
pks login -a PKS-API -u USERNAME -p PASSWORD --ca-cert CERT-PATH
```

Replace the placeholder values in the command as follows:

- `PKS-API` is the domain name for the PKS API that you entered in **Ops Manager** > **Pivotal Container Service** > **PKS API** > **API Hostname (FQDN)**. For example, `api.pks.example.com`.
- `USERNAME` and `PASSWORD` belong to the account you created in the Grant PKS Access to a User section of *Managing Users in PKS with UAA*. If you do not use `-p` to provide a password, the PKS CLI prompts for the password interactively. Pivotal recommends running the login command without the `-p` flag for added security.
- `CERT-PATH` is the path to your root CA certificate. Provide the certificate to validate the PKS API certificate with SSL.

  For example:

  ```
  $ pks login -a api.pks.example.com -u alana \
  --ca-cert /var/tempest/workspaces/default/root_ca_certificate
  ```

  If you are logging in to a trusted environment, you can use `-k` to skip SSL verification instead of `--ca-cert CERT-PATH`.

  For example:

  ```
  $ pks login -a api.pks.example.com -u alana -k
  ```

Please send any feedback you have to pks-feedback@pivotal.io.

Accessing Dashboard

This topic describes how to access Dashboard, a web-based Kubernetes UI, for your Pivotal Container Service (PKS) deployment.

Overview

Kubernetes provides Dashboard to manage Kubernetes clusters and applications, and to review the state of Kubernetes cluster resources.

Access Credentials

You must have either a `kubectl` Kubeconfig or Bearer Token access credential to access Dashboard.

Configure Kubeconfig Access Credentials

You can use the PKS CLI to request a Kubeconfig access credential and to save the credential to either a file or environment variable for use as your Dashboard access credential.

To request Kubeconfig credentials use one of the two following methods.

- Request a Kubeconfig access credential using the PKS CLI:

  ```
  pks get-credentials CLUSTER-NAME
  ```

  Where `CLUSTER-NAME` is the name of your cluster.

  For example:

  ```
  $ pks get-credentials pks-bosh


  Fetching credentials for cluster pks-bosh.
  Context set for cluster pks-bosh.
  ```

- Request a Kubeconfig access credential and assign to your Kubernetes configuration:

  ```
  KUBECONFIG=CONFIG-FILE pks get-credentials CLUSTER-NAME
  ```

  Where:

  - `CONFIG-FILE` is the name of the output file which will store the exported access credentials.
  - `CLUSTER-NAME` is the name of your cluster.

Request Bearer Token Access Credentials

You can use `kubectl` to request a Bearer Token access credential.

1. To request your Kubernetes user ID, run the following command:

   ```
   kubectl config view -o jsonpath='{.contexts[?(@.name == "CLUSTER-NAME")].context.user}'
   ```

   Where `CLUSTER-NAME` is the name of your cluster.

   For example:

   ```
   $ kubectl config view -o jsonpath='{.contexts[?(@.name == "pks-bosh")].context.user}'
   dxbjlm0j-ac11-43f9-99a7-87u5u4fbe44b
   ```

2. To derive a Kubeconfig Token use one of the two following methods.

- Kubectl Get Secret request:

  ```
  kubectl describe secret $(kubectl get secret | grep USER-ID | awk '{print $1}') | grep "token:"
  ```

  Where `USER-ID` is your Kubernetes User ID.

  For example:

  ```
  $ kubectl describe secret $(kubectl get secret | grep dxbjlm0j-ac11-43f9-99a7-87u5u4fbe44b | awk '{print $1}') | grep "token:"
  token:      eyxYzGciOiJSUzI1NiPsIndxbaac0jac11erf999a787e3e4fbe44rgnZ....iI4utgU6-qKDEdwEJw5TQA
  ```

- Kubectl Describe Service Accounts request:

  ```
  kubectl describe secret $(kubectl describe serviceaccounts USER-ID | grep Tokens | awk '{print $2}') | grep "token:"
  ```

  Where `USER-ID` is your Kubernetes User ID.

  For example:

  ```
  $ kubectl describe secret $(kubectl describe serviceaccounts dxbjlm0j-ac11-43f9-99a7-87u5u4fbe44b | grep Tokens | awk '{print $2}') | grep "token:"
  token:      eyxYzGciOiJSUzI1NiPsIndxbaac0jac11erf999a787e3e4fbe44rgnZ....iI4utgU6-qKDEdwEJw5TQA
  ```

Access Dashboard

After you have obtained access credentials you can authenticate into Dashboard.

1. To start the proxy server run the following:

   ```
   kubectl proxy
   ```

2. To access the Dashboard UI, open a browser and navigate to the following:

   ```
   http://localhost:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/
   ```

3. On the Kubernetes Dashboard sign in page select an option based on the type of credential that you prepared in the previous steps.

   - If you prepared a Kubeconfig credential file:

     - Select **Kubeconfig**.
     - To specify your kubeconfig file select `...`, to the right of **Choose kubeconfig file**.
     - Specify the kubeconfig file location.

   - If you prepared a Kubeconfig token:

     - Select **Token**.
     - To specify your kuebconfig token, paste your kuebconfig token into the **Enter token** area.

4. Click **SIGN IN**. The Dashboard Overview page is displayed.

Use Dashboard

For information about how to use Dashboard, see Web UI (Dashboard) in the Kubernetes documentation.

Please send any feedback you have to pks-feedback@pivotal.io.

Deploying and Exposing Basic Workloads

Page last updated:

This topic describes how to configure, deploy, and expose basic workloads in Pivotal Container Service (PKS).

Overview

A load balancer is a third-party device that distributes network and application traffic across resources. Using a load balancer can prevent individual network components from being overloaded by high traffic.

Note: The procedures in this topic create a dedicated load balancer for each workload. If your cluster has many apps, a load balancer dedicated to each workload can be an inefficient use of resources. An ingress controller pattern is better suited for clusters with many workloads.

Refer to the following PKS documentation topics for additional information about deploying and exposing workloads:

- For the different types of load balancers used in a deployment, see Load Balancers in PKS.
- For ingress routing on GCP, AWS, Azure, or vSphere without NSX-T, see Configuring Ingress Routing.
- For ingress routing on vSphere with NSX-T, see Configuring Ingress Resources and Load Balancer Services.

Prerequisites

This topic references standard Kubernetes primitives. If you are unfamiliar with Kubernetes primitives, review the Kubernetes Workloads and Services, Load Balancing, and Networking documentation before following the procedures below.

vSphere without NSX-T Prerequisites

If you use vSphere without NSX-T, you can choose to configure your own external load balancer or expose static ports to access your workload without a load balancer. See Deploy Workloads without a Load Balancer below.

GCP, AWS, Azure, and vSphere with NSX-T Prerequisites

If you use Google Cloud Platform (GCP), Amazon Web Services (AWS), Azure, or vSphere with NSX-T integration, your cloud provider can configure a public-cloud external load balancer for your workload. See either Deploy Workloads on vSphere with NSX-T or Deploy Workloads on GCP, AWS, or Azure, Using a Public-Cloud External Load Balancer below.

AWS Prerequisites

If you use AWS, you can also expose your workload using a public-cloud internal load balancer.

Perform the following steps before you create a load balancer:

1. In the AWS Management Console, create or locate a public subnet for each availability zone (AZ) that you are deploying to. A public subnet has a route table that directs internet-bound traffic to the internet gateway.
2. On the command line, run `pks cluster CLUSTER-NAME`, where `CLUSTER-NAME` is the name of your cluster.
3. Record the unique identifier for the cluster.
4. In the AWS Management Console, tag each public subnet based on the table below, replacing `CLUSTER-UUID` with the unique identifier of the cluster. Leave the **Value** field empty.

Key	Value
`kubernetes.io/cluster/service-instance_CLUSTER-UUID`	empty

Note: AWS limits the number of tags on a subnet to 100.

After completing these steps, follow the steps below in Deploy AWS Workloads Using an Internal Load Balancer.

Deploy Workloads on vSphere with NSX-T

If you use vSphere with NSX-T, follow the steps below to deploy and expose basic workloads using the NSX-T load balancer.

Configure Your Workload

1. Open your workload's Kubernetes service configuration file in a text editor.
2. To expose the workload through a load balancer, confirm that the Service object is configured to be `type: LoadBalancer`.

   For example:

   ```
   ---
   apiVersion: v1
   kind: Service
   metadata:
     labels:
       name: nginx
     name: nginx
   spec:
     ports:
       - port: 80
     selector:
       app: nginx
     type: LoadBalancer
   ---
   ```

3. Confirm the workload's Kubernetes service configuration is set to be `type: LoadBalancer`.
4. Confirm the `type` property of each workload's Kubernetes service is similarly configured.

Note: For an example of a fully configured Kubernetes service, see the nginx app's example `type: LoadBalancer` configuration in GitHub.

For more information about configuring the `LoadBalancer` Service type see the Kubernetes documentation.

Deploy and Expose Your Workload

1. To deploy the service configuration for your workload, run the following command:

   ```
   kubectl apply -f SERVICE-CONFIG
   ```

   Where `SERVICE-CONFIG` is your workload's Kubernetes service configuration.

   For example:

   ```
   kubectl apply -f nginx.yml
   ```

   This command creates three pod replicas, spanning three worker nodes.

2. Deploy your applications, deployments, config maps, persistent volumes, secrets, and any other configurations or objects necessary for your applications to run.
3. Wait until your cloud provider has created and connected a dedicated load balancer to the worker nodes on a specific port.

Access Your Workload

1. To determine your exposed workload's load balancer IP address and port number, run the following command:

```
kubectl get svc SERVICE-NAME
```

Where `SERVICE-NAME` is your workload configuration's specified service `name`.

For example:

```
kubectl get svc nginx
```

2. Retrieve the load balancer's external IP address and port from the returned listing.
3. To access the app, run the following on the command:

```
curl http://EXTERNAL-IP:PORT
```

Where:

- `EXTERNAL-IP` is the IP address of the load balancer
- `PORT` is the port number.

Note: This command should be run on a server with network connectivity and visibility to the IP address of the worker node.

Deploy Workloads on GCP, AWS, or Azure, Using a Public-Cloud External Load Balancer

If you use GCP, AWS, or Azure, follow the steps below to deploy and expose basic workloads using a load balancer configured by your cloud provider.

Configure Your Workload

1. Open your workload's Kubernetes service configuration file in a text editor.
2. To expose the workload through a load balancer, confirm that the Service object is configured to be `type: LoadBalancer`.

   For example:

```
---
apiVersion: v1
kind: Service
metadata:
  labels:
    name: nginx
  name: nginx
spec:
  ports:
    - port: 80
  selector:
    app: nginx
  type: LoadBalancer
---
```

3. Confirm the workload's Kubernetes service configuration is set to be `type: LoadBalancer`.
4. Confirm the `type` property of each workload's Kubernetes service is similarly configured.

Note: For an example of a fully configured Kubernetes service, see the nginx app's example `type: LoadBalancer` configuration in GitHub.

For more information about configuring the `LoadBalancer` Service type see the Kubernetes documentation.

Deploy and Expose Your Workload

1. To deploy the service configuration for your workload, run the following command:

```
kubectl apply -f SERVICE-CONFIG
```

Where `SERVICE-CONFIG` is your workload's Kubernetes service configuration.

For example:

```
kubectl apply -f nginx.yml
```

This command creates three pod replicas, spanning three worker nodes.

2. Deploy your applications, deployments, config maps, persistent volumes, secrets, and any other configurations or objects necessary for your applications to run.

3. Wait until your cloud provider has created and connected a dedicated load balancer to the worker nodes on a specific port.

Access Your Workload

1. To determine your exposed workload's load balancer IP address and port number, run the following command:

```
kubectl get svc SERVICE-NAME
```

Where `SERVICE-NAME` is your workload configuration's specified service `name`.

For example:

```
kubectl get svc nginx
```

2. Retrieve the load balancer's external IP address and port from the returned listing.

3. To access the app, run the following on the command:

```
curl http://EXTERNAL-IP:PORT
```

Where:

- `EXTERNAL-IP` is the IP address of the load balancer
- `PORT` is the port number.

Note: This command should be run on a server with network connectivity and visibility to the IP address of the worker node.

Deploy AWS Workloads Using an Internal Load Balancer

If you use AWS, follow the steps below to deploy, expose, and access basic workloads using an internal load balancer configured by your cloud provider.

Configure Your Workload

1. Open your workload's Kubernetes service configuration file in a text editor.

2. To expose the workload through a load balancer, confirm that the Service object is configured to be `type: LoadBalancer`.

3. In the services metadata section of the manifest, add the following `annotations` tag:

```
annotations:
    service.beta.kubernetes.io/aws-load-balancer-internal: 0.0.0.0/0
```

For example:

```
---
apiVersion: v1
kind: Service
metadata:
 labels:
   name: nginx
 annotations:
      service.beta.kubernetes.io/aws-load-balancer-internal: 0.0.0.0/0
 name: nginx
spec:
 ports:
   - port: 80
 selector:
   app: nginx
 type: LoadBalancer
 ---
```

4. Confirm that the workload's Kubernetes service configuration is set to be `type: LoadBalancer`.
5. Confirm that the `annotations` and `type` properties of each workload's Kubernetes service are similarly configured.

Note: For an example of a fully configured Kubernetes service, see the nginx app's example `type: LoadBalancer` configuration in GitHub.

For more information about configuring the `LoadBalancer` Service type see the Kubernetes documentation.

Deploy and Expose Your Workload

1. To deploy the service configuration for your workload, run the following command:

   ```
   kubectl apply -f SERVICE-CONFIG
   ```

   Where `SERVICE-CONFIG` is your workload's Kubernetes service configuration.
   For example:

   ```
   kubectl apply -f nginx.yml
   ```

   This command creates three pod replicas, spanning three worker nodes.

2. Deploy your applications, deployments, config maps, persistent volumes, secrets, and any other configurations or objects necessary for your applications to run.
3. Wait until your cloud provider has created and connected a dedicated load balancer to the worker nodes on a specific port.

Access Your Workload

1. To determine your exposed workload's load balancer IP address and port number, run the following command:

   ```
   kubectl get svc SERVICE-NAME
   ```

   Where `SERVICE-NAME` is your workload configuration's specified service `name`.
   For example:

   ```
   kubectl get svc nginx
   ```

2. Retrieve the load balancer's external IP and port from the returned listing.
3. To access the app, run the following command:

   ```
   curl http://EXTERNAL-IP:PORT
   ```

   Where:

   - `EXTERNAL-IP` is the IP address of the load balancer.
   - `PORT` is the port number.

Note: This command should be run on a server with network connectivity and visibility to the IP address of the worker node.

Deploy Workloads for a Generic External Load Balancer

Follow the steps below to deploy and access basic workloads using a generic external load balancer, such as F5.

In this approach you will access you workloads with a generic external load balancer.

Using a generic external load balancer requires a static port in your Kubernetes cluster. To do this we need to expose your workloads with a `NodePort`.

Configure Your Workload

To expose a static port on your workload, perform the following steps:

1. Open your workload's Kubernetes service configuration file in a text editor.
2. To expose the workload without a load balancer, confirm that the Service object is configured to be `type: NodePort`.
   For example:

```
---
apiVersion: v1
kind: Service
metadata:
  labels:
    name: nginx
  name: nginx
spec:
  ports:
    - port: 80
  selector:
    app: nginx
  type: NodePort
---
```

3. Confirm that the workload's Kubernetes service configuration is set to be `type: NodePort`.
4. Confirm that the `type` property of each workload's Kubernetes service is similarly configured.

Note: For an example of a fully configured Kubernetes service, see the nginx app's example `type: NodePort` configuration in GitHub.

For more information about configuring the `NodePort` Service type see the Kubernetes documentation.

Deploy and Expose Your Workload

1. To deploy the service configuration for your workload, run the following command:

```
kubectl apply -f SERVICE-CONFIG
```

   Where `SERVICE-CONFIG` is your workload's Kubernetes service configuration.
   For example:

```
kubectl apply -f nginx.yml
```

   This command creates three pod replicas, spanning three worker nodes.

2. Deploy your applications, deployments, config maps, persistent volumes, secrets, and any other configurations or objects necessary for your applications to run.
3. Wait until your cloud provider has connected your worker nodes on a specific port.

Access Your Workload

1. Retrieve the IP address for a worker node with a running app pod.

   Note: If you deployed more than four worker nodes, some worker nodes may not contain a running app pod. Select a worker node that contains a running app pod.

   You can retrieve the IP address for a worker node with a running app pod in one of the following ways:

   - On the command line, run the following

   ```
   kubectl get nodes -L spec.ip
   ```

   - On the Ops Manager command line, run the following to find the IP address:

   ```
   bosh vms
   ```

   This IP address will be used when configuring your external load balancer.

2. To see a listing of port numbers, run the following command:

   ```
   kubectl get svc SERVICE-NAME
   ```

   Where `SERVICE-NAME` is your workload configuration's specified service `name`.
   For example:

   ```
   kubectl get svc nginx
   ```

3. Find the node port number in the `3XXXX` range. This port number will be used when configuring your external load balancer.

4. Configure your external load balancer to map your application Uri to the IP and port number you collected above. Please refer to your load balancer documentation for instructions.

Deploy Workloads without a Load Balancer

If you do not use an external load balancer, you can configure your service to expose a static port on each worker node. The following steps configure your service to be reachable from outside the cluster at `http://NODE-IP:NODE-PORT`.

Configure Your Workload

To expose a static port on your workload, perform the following steps:

1. Open your workload's Kubernetes service configuration file in a text editor.

2. To expose the workload without a load balancer, confirm that the Service object is configured to be `type: NodePort`.
   For example:

   ```
   ---
   apiVersion: v1
   kind: Service
   metadata:
     labels:
       name: nginx
     name: nginx
   spec:
     ports:
       - port: 80
     selector:
       app: nginx
     type: NodePort
   ---
   ```

3. Confirm that the workload's Kubernetes service configuration is set to be `type: NodePort`.

4. Confirm that the `type` property of each workload's Kubernetes service is similarly configured.

Note: For an example of a fully configured Kubernetes service, see the nginx app's example `type: NodePort` configuration in GitHub.

For more information about configuring the `NodePort` Service type see the Kubernetes documentation.

Deploy and Expose Your Workload

1. To deploy the service configuration for your workload, run the following command:

    ```
    kubectl apply -f SERVICE-CONFIG
    ```

    Where `SERVICE-CONFIG` is your workload's Kubernetes service configuration.
    For example:

    ```
    kubectl apply -f nginx.yml
    ```

    This command creates three pod replicas, spanning three worker nodes.

2. Deploy your applications, deployments, config maps, persistent volumes, secrets, and any other configurations or objects necessary for your applications to run.

3. Wait until your cloud provider has connected your worker nodes on a specific port.

Access Your Workload

1. Retrieve the IP address for a worker node with a running app pod.

    Note: If you deployed more than four worker nodes, some worker nodes may not contain a running app pod. Select a worker node that contains a running app pod.

    You can retrieve the IP address for a worker node with a running app pod in one of the following ways:

    - On the command line, run the following

    ```
    kubectl get nodes -L spec.ip
    ```

    - On the Ops Manager command line, run the following to find the IP address:

    ```
    bosh vms
    ```

2. To see a listing of port numbers, run the following command:

    ```
    kubectl get svc SERVICE-NAME
    ```

    Where `SERVICE-NAME` is your workload configuration's specified service `name`.
    For example:

    ```
    kubectl get svc nginx
    ```

3. Find the node port number in the `3XXXX` range.

4. To access the app, run the following command line:

    ```
    curl http://NODE-IP:NODE-PORT
    ```

    Where

    - `NODE-IP` is the IP address of the worker node.
    - `NODE-PORT` is the node port number.

    Note: Run this command on a server with network connectivity and visibility to the IP address of the worker node.

Please send any feedback you have to pks-feedback@pivotal.io.

Getting Started with VMware Harbor Registry

This topic describes VMware Harbor Registry, an enterprise-class image registry server that stores and distributes container images for Pivotal Container Service (PKS).

Overview

Harbor allows you to store and manage container images for your PKS deployment. Deploying an image registry alongside PKS improves image transfer speed.

As an enterprise private registry, Harbor also offers enhanced performance and improved security. By configuring Harbor with PKS, you can apply enterprise features to your image registry, such as security, identity, and management.

You can install Harbor alongside PKS on vSphere, Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure.

Install Harbor

To install Harbor, do the following:

1. Install PKS. See the *Installing PKS* topic for your cloud provider.
2. Install Harbor. See Installing and Configuring VMware Harbor Registry.

Use Harbor

Before you can push images to Harbor, you must do the following:

1. Configure authentication and role-based access control (RBAC) for Harbor. See Role Based Access Control (RBAC) in the Harbor User Guide on GitHub.
2. Create a Harbor project that contains all repositories for your app. See Managing projects in the Harbor User Guide on GitHub.

After you configure Harbor, you can do the following:

- Push or pull Docker images to your Harbor project using the Docker command-line interface (CLI). See Pulling and pushing images using Docker client in the Harbor User Guide on GitHub.
- Manage Helm charts in your Harbor project using either the Harbor portal or the Helm CLI. See Manage Helm Charts in the Harbor User Guide on GitHub.
- Install Clair to enable vulnerability scanning for images stored in Harbor. See Step 8: Configure Container Vulnerability Scanning Using Clair in *Installing and Configuring VMware Harbor Registry*.

For more information about managing images in Harbor, see the User Guide in the Harbor repository on GitHub.

Manage Harbor

As a Harbor administrator, you can manage the following in the Harbor portal:

- **Authentication**: Select either local user authentication or configure LDAP/Active Directory integration. If you select local user authentication, you can enable or disable user self-registration.
- **Users and roles**: Manage privileges for Harbor users.
- **Email settings**: Configure a mail server for user password resets.
- **Project creation**: Specify which users can create projects.
- **Registry permissions**: Manage permissions for image registry access.
- **Endpoints**: Add and remove image registry endpoints.
- **Replication policies**: Add and remove rules for replication jobs.

For more information about managing Harbor as an administrator, see Administrator options in the Harbor User Guide on GitHub.

Please send any feedback you have to pks-feedback@pivotal.io.

Using Helm with PKS

Page last updated:

This topic describes how to use the package manager Helm for your Kubernetes apps running on Pivotal Container Service (PKS).

Overview

Helm includes the following components:

Component	Role	Location
`helm`	Client	Runs on your local workstation
`tiller`	Server	Runs inside your Kubernetes cluster

Helm packages are called **charts**. For more information, see Charts in the Helm documentation.

Examples of charts:

- Concourse for CI/CD pipelines
- Datadog for monitoring
- MySQL for storage

For more charts, see the Helm Charts repository on GitHub.

Configure Tiller

If you want to use Helm with PKS, you must configure Tiller.

Tiller runs inside the Kubernetes cluster and requires access to the Kubernetes API.

If you use role-based access control (RBAC) in PKS, perform the steps in this section to grant Tiller permission to access the API.

1. Create a file named `rbac-config.yaml` with the following configuration:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: tiller
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: tiller
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
  - kind: ServiceAccount
    name: tiller
    namespace: kube-system
```

2. Create the service account and role by running the following command:

```
kubectl create -f rbac-config.yaml
```

3. Download and install the Helm CLI.

4. Deploy Helm using the service account by running the following command:

```
helm init --service-account tiller
```

5. Verify that the permissions are configured by running the following command:

    ```
    helm ls
    ```

    There should be no output from the above command.

To apply more granular permissions to the Tiller service account, see the Helm RBAC documentation.

Please send any feedback you have to pks-feedback@pivotal.io.

Configuring and Using PersistentVolumes

Page last updated:

This topic describes how to provision static and dynamic PersistentVolumes (PVs) for Pivotal Container Service (PKS) to run stateful apps.

For static PV provisioning, the PersistentVolumeClaim (PVC) does not need to reference a StorageClass. For dynamic PV provisioning, you must specify a StorageClass and define the PVC using a reference to that StorageClass.

For more information about storage management in Kubernetes, refer to the Kubernetes documentation.

For more information about the supported vSphere topologies for PV storage, see PersistentVolume Storage Options on vSphere.

Provision a Static PV

To provision a static PV, you manually create a Virtual Machine Disk (VMDK) file to use as a storage backend for the PV. When the PV is created, Kubernetes knows which volume instance is ready for use. When a PVC or volumeClaimTemplate is requested, Kubernetes chooses an available PV in the system and allocates it to the Deployment or StatefulSets workload.

Provision a Static PV for a Deployment Workload

To provision a static PV for a Deployment workload, the procedure is as follows:

Note: The examples in this section use the vSphere volume plugin. Refer to the Kubernetes documentation for information about volume plugins for other cloud providers.

1. `ssh` into an ESXi host in your vCenter cluster that has access to the datastore where you will host the static PV.
2. Create VMDK files, replacing `DATASTORE` with your datastore directory name:

```
[root@ESXi-1:~] cd /vmfs
[root@ESXi-1:/vmfs] cd volumes/
[root@ESXi-1:/vmfs/volumes] cd DATASTORE/
[root@ESXi-1:/vmfs/volumes/7e6c0ca3-8c4873ed] cd kubevols/
[root@ESXi-1:/vmfs/volumes/7e6c0ca3-8c4873ed/kubevols] vmkfstools -c 2G redis-master.vmdk
```

3. Define a PV using a YAML manifest file that contains a reference to the VMDK file. For example, on vSphere, create a file named `redis-master-pv.yaml` with the following contents:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: redis-master-pv
spec:
  capacity:
    storage: 2Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  vsphereVolume:
    volumePath: "[NFS-LAB-DATASTORE] kubevols/redis-master"
    fsType: ext4
```

4. Define a PVC using a YAML manifest file. For example, create a file named `redis-master-claim.yaml` with the following contents:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: redis-master-claim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
```

5. Define a deployment using a YAML manifest file that references the PVC. For example, create a file named `redis-master.yaml` with the following contents:

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: redis-master
…
spec:
  template:
    spec:
      volumes:
      - name: redis-master-data
      persistentVolumeClaim:
        claimName: redis-master-claim
```

Provision a Static PV for a StatefulSets Workload

To provision a static PV for a StatefulSets workload with three replicas, the procedure is as follows:

Note: The examples in this section use the vSphere volume plugin. Refer to the Kubernetes documentation for information about volume plugins for other cloud providers.

1. Create VMDK files, replacing `DATASTORE` with your datastore directory name:

```
[root@ESXi-1:~] cd /vmfs
[root@ESXi-1:/vmfs] cd volumes/
[root@ESXi-1:/vmfs/volumes] cd DATASTORE/
[root@ESXi-1:/vmfs/volumes/7e6c0ca3-8c4873ed] cd kubevols/
[root@ESXi-1:/vmfs/volumes/7e6c0ca3-8c4873ed/kubevols] vmkfstools -c 10G mysql-pv-1.vmdk
[root@ESXi-1:/vmfs/volumes/7e6c0ca3-8c4873ed/kubevols] vmkfstools -c 10G mysql-pv-2.vmdk
[root@ESXi-1:/vmfs/volumes/7e6c0ca3-8c4873ed/kubevols] vmkfstools -c 10G mysql-pv-3.vmdk
```

2. Define a PV for the first replica using a YAML manifest file that contains a reference to the VMDK file. For example, on vSphere, create a file named `mysql-pv-1.yaml` with the following contents:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mysql-pv-1
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  vsphereVolume:
    volumePath: "[NFS-LAB-DATASTORE] kubevols/mysql-pv-1"
    fsType: ext4
```

3. Define a PV for the second replica using a YAML manifest file that contains a reference to the VMDK file. For example, on vSphere, create a file named `mysql-pv-2.yaml` with the following contents:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mysql-pv-2
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  vsphereVolume:
    volumePath: "[NFS-LAB-DATASTORE] kubevols/mysql-pv-2"
    fsType: ext4
```

4. Define a PV for the third replica using a YAML manifest file that contains a reference to the VMDK file. For example, on vSphere, create a file named `mysql-pv-3.yaml` with the following contents:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mysql-pv-3
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  vsphereVolume:
    volumePath: "[NFS-LAB-DATASTORE] kubevols/mysql-pv-3"
    fsType: ext4
```

5. Define a StatefultSets object using a YAML manifest file. For example, create a file named `mysql-statefulsets.yaml` with the following contents:

```
piVersion: apps/v1
kind: StatefulSet
metadata:
  name: mysql
spec:
  selector:
    matchLabels:
      app: mysql
  serviceName: mysql
  replicas: 3
...
volumeClaimTemplates:
  - metadata:
    name: data
    spec:
    accessModes: ["ReadWriteOnce"]
    resources:
       requests:
       storage: 10Gi
```

Note: In previous steps you created a total of three PVs. The `spec.replicas: 3` field defines three replicas. Each replica is attached to one PV.

Note: In the volumeClaimTemplates section, you must specify the required storage size for each replica. Do not to refer to a StorageClass.

Provision a Dynamic PV

Dynamic PV provisioning gives developers the freedom to provision storage when they need it without manual intervention from a Kubernetes cluster administrator. To enable dynamic PV provisioning, the Kubernetes cluster administrator defines one or more StorageClasses.

For dynamic PV provisioning, the procedure is to define and create a PVC that automatically triggers the creation of the PV and its backend VMDK file. When the PV is created, Kubernetes knows which volume instance is available for use. When a PVC or volumeClaimTemplate is requested, Kubernetes chooses an available PV and allocates it to the Deployment or StatefulSets workload.

PKS supports dynamic PV provisioning by providing StorageClasses for all supported cloud providers, as well as an example PVC.

Note: For dynamic PVs on vSphere, you must create or map the VMDK file for the StorageClass on a shared file system datastore. This shared file system datastore must be accessible to each vSphere cluster where Kubernetes cluster nodes run. For more information, see PersistentVolume Storage Options on vSphere.

Provision a Dynamic PV for Deployment Workloads

Note: The examples in this section use the vSphere provisioner. Refer to the Kubernetes documentation for information about provisioners for other cloud providers.

For the Deployment workload with dynamic PV provisioning, the procedure is as follows:

1. Define a StorageClass using a YAML manifest file. For example, on vSphere, create a file named `redis-sc.yaml` with the following contents:

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: thin-disk
provisioner: kubernetes.io/vsphere-volume
parameters:
    datastore: Datastore-NFS-VM
    diskformat: thin
    fstype: ext3
```

2. Define a PVC using a YAML manifest file that references the StorageClass. For example, create a file named `redis-master-claim.yaml` with the following contents:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: redis-master-claim
  annotations:
    volume.beta.kubernetes.io/storage-class: thin-disk
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
    storage: 2Gi
```

Note: When you deploy the PVC on vSphere, the vSphere Cloud Provider plugin automatically creates the PV and associated VMDK file.

3. Define a Deployment using a YAML manifest file that references the PVC. For example, create a file named `redis-master.yaml` with the following contents:

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: redis-master
…
spec:
  template:
    spec:
      volumes:
      - name: redis-master-data
        persistentVolumeClaim:
        claimName: redis-master-claim
```

Provision a Dynamic PV for StatefulSets Workloads

Note: The examples in this section use the vSphere provisioner. Refer to the Kubernetes documentation for information about provisioners for other cloud providers.

To provision a static PV for a StatefulSets workload with three replicas, the procedure is as follows:

1. Define a StorageClass using a YAML manifest file. For example, on vSphere, create a file named `mysql-sc.yaml` with the following contents:

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: my-storage-class
provisioner: kubernetes.io/vsphere-volume
parameters:
    datastore: Datastore-NFS-VM
    diskformat: thin
    fstype: ext3
```

2. Define a StatefulSets object using a YAML manifest file that references the StorageClass. For example, create a file named `mysql-statefulsets.yaml` with the following contents:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: mysql
spec:
...
volumeClaimTemplates:
  - metadata:
    name: data
    spec:
    accessModes: ["ReadWriteOnce"]
      storageClassName: "my-storage-class"
    resources:
       requests:
       storage: 10Gi
```

Note: In the volumeClaimTemplates, specify the required storage size for each replica. Unlike static provisioning, you must explicitly refer to the desired StorageClass when you use dynamic PV provisioning.

Specify a Default StorageClass

If you have or anticipate having more than one StorageClass for use with dynamic PVs for a Kubernetes cluster, you may want to designate a particular StorageClass as the default. This allows you to manage a storage volume without setting up specialized StorageClasses across the cluster.

If necessary, a developer can change the default StorageClass in the PVC definition. See the Kubernetes documentation for more information.

To specify a StorageClass as the default for a Kubernetes cluster, use the annotation `storageclass.kubernetes.io/is-default-class: "true"`.

For example:

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: thin-disk
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: kubernetes.io/vsphere-volume
parameters:
    datastore: Datastore-NFS-VM
    diskformat: thin
    fstype: ext3
```

Note: The above example uses the vSphere provisioner. Refer to the Kubernetes documentation for information about provisioners for other cloud providers.

Provision Dynamic PVs for Use with PKS

Perform the steps in this section to register one or more StorageClasses and define a PVC that can be applied to newly-created pods.

1. Download the StorageClass spec for your cloud provider.
    - **AWS**:

      ```
      $ wget https://raw.githubusercontent.com/cloudfoundry-incubator/kubo-ci/master/specs/storage-class-aws.yml
      ```

    - **Azure**:
        - For Azure disk storage:

          ```
          $ wget https://raw.githubusercontent.com/cloudfoundry-incubator/kubo-ci/master/specs/storage-class-azure.yml
          ```

        - For Azure file storage:

          ```
          $ wget https://raw.githubusercontent.com/cloudfoundry-incubator/kubo-ci/master/specs/storage-class-azure-file.yml
          ```

- **GCP**:

  ```
  $ wget https://raw.githubusercontent.com/cloudfoundry-incubator/kubo-ci/master/specs/storage-class-gcp.yml
  ```

- **vSphere**:

  ```
  $ wget https://raz.githubusercontent.com/cloudfoundry-incubator/kubo-ci/master/specs/storage-class-vsphere.yml
  ```

  After downloading the vSphere StorageClass spec, replace the contents of the file with the following YAML to create the correct StorageClass for vSphere:

  ```
  kind: StorageClass
  apiVersion: storage.k8s.io/v1
  metadata:
    name: thin
    annotations:
      storageclass.kubernetes.io/is-default-class: "true"
  provisioner: kubernetes.io/vsphere-volume
  parameters:
      diskformat: thin
  ```

2. Apply the spec by running `kubectl create -f STORAGE-CLASS-SPEC.yml`. Replace `STORAGE-CLASS-SPEC` with the name of the file you downloaded in the previous step.

   For example:

   ```
   $ kubectl create -f storage-class-gcp.yml
   ```

3. Run the following command to download the example PVC:

   ```
   $ wget https://raw.githubusercontent.com/cloudfoundry-incubator/kubo-ci/master/specs/persistent-volume-claim.yml
   ```

4. Run the following command to apply the PVC:

   ```
   $ kubectl create -f persistent-volume-claim.yml
   ```

   - To confirm you applied the PVC, run the following command:

     ```
     $ kubectl get pvc -o wide
     ```

5. To use the dynamic PV, create a pod that uses the PVC. See the pv-guestbook.yml configuration file as an example.

Logging out of PKS

On the command line, run `pks logout` to log out of your PKS environment.

After logging out, you must run `pks login` before you can run any other `pks` commands.

Please send any feedback you have to pks-feedback@pivotal.io.

Logging and Monitoring PKS

This section describes how to monitor Pivotal Container Service (PKS) deployments.

See the following topics:

- Viewing Usage Data
- Downloading Cluster Logs
- Monitoring PKS with Sinks
- Monitoring Master/etcd Node VMs

For information about monitoring PKS with VMware Wavefront, see VMware PKS Integration.

Please send any feedback you have to pks-feedback@pivotal.io.

Viewing Usage Data

Page last updated:

This topic describes how operators can view pod usage information from their Pivotal Container Service (PKS) deployment. Operators can use this data to calculate billed usage, perform customer chargebacks, and generate usage reports.

The PKS database stores the following pod usage data:

- **Watermark**: the number of pods that run at a single time.
- **Consumption**: the memory and CPU usage of pods.

About Usage Data

This section describes the usage data records you can view in the PKS billing database. The agent pod collects usage data for the deployment and sends the data to the PKS aggregator agent. The aggregator agent then stores the data in the PKS database. You can access the PKS database from the PKS VM.

The following is an example of a pod usage data table:

```
+--------------------------------------+---------------------+---------------------+-------------+-------------+-----------------------------------------------------+
| id                                   | first_seen          | last_seen           | namespace   | name        | service_instance_id                                 |
+--------------------------------------+---------------------+---------------------+-------------+-------------+-----------------------------------------------------+
| 12a345b6-7890-13c4-de5f-67890a123b4c | 2019-01-07 13:57:03 | 2019-01-08 11:34:33 | my-namespace | my-pod      | service-instance_a12b3456-78cd-90e1-fa2b-3456c789def0 |
| ac203f27-104b-11e9-b520-42010a000b0a | 2019-01-04 18:09:04 | 2019-01-07 14:09:03 | my-namespace | my-other-pod | service-instance_a12b3456-78cd-90e1-fa2b-3456c789def0 |
+--------------------------------------+---------------------+---------------------+-------------+-------------+-----------------------------------------------------+
2 rows in set (0.00 sec)
```

The following table describes the fields that appear in the pod usage data table:

Field Name	Description
id	Unique record identifier
first_seen	The date when the pod was first recorded to the database
last_seen	The date when the pod was most recently recorded to the database
namespace	The namespace where the pod is deployed
name	The name of the pod
service_instance_id	The cluster where the pod is deployed

View Usage Data

To view the pod usage data table, follow the steps below:

1. In a browser, navigate to Ops Manager.
2. Select the **Pivotal Container Service** tile.
3. Select the **Status** tab. Record the IP address that appears in the **IPS** column.
4. Select the **Credentials** tab.
5. Click the credential link next to **Cf Mysql Billing Db Password**. Record the billing database password that appears.
6. Open a terminal window from any system inside your PKS network. If your system is outside the network, you can SSH into the PKS VM. For more information, see SSH into the PKS API VM.
7. On the command line, log in to the billing database by running `mysql -h IP-ADDRESS -u billing -p billing`, replacing `IP-ADDRESS` with the IP you located in a previous step.

   For example:

   ```
   mysql -h 10.0.10.10 -u billing -p billing
   ```

8. When prompted by the command line, enter the billing database password you recorded in a previous step.

9. View the tables in the billing database by running `show tables;`.

   For example:

```
MariaDB [billing]> show tables;
+------------------+
| Tables_in_billing |
+------------------+
| pods             |
| schema_migrations |
+------------------+
2 rows in set (0.00 sec)
```

10. View the raw pod usage data in the `pods` table by running `select * from pods;`.

    For example:

```
MariaDB [billing]> select * from pods;


+--------------------------------------+---------------------+---------------------+-------------+--------------+--------------------------------------------------------+
| id                                   | first_seen          | last_seen           | namespace   | name         | service_instance_id                                    |
+--------------------------------------+---------------------+---------------------+-------------+--------------+--------------------------------------------------------+
| 12a345b6-7890-13c4-de5f-67890a123b4c | 2019-01-07 13:57:03 | 2019-01-08 11:34:33 | my-namespace | my-pod       | service-instance_a12b3456-78cd-90e1-fa2b-3456c789def0 |
| ac203f27-104b-11e9-b520-42010a000b0a | 2019-01-04 18:09:04 | 2019-01-07 14:09:03 | my-namespace | my-other-pod | service-instance_a12b3456-78cd-90e1-fa2b-3456c789def0 |
+--------------------------------------+---------------------+---------------------+-------------+--------------+--------------------------------------------------------+
2 rows in set (0.00 sec)
```

11. (Optional) For information about running additional queries against the billing database, see the following articles in the Pivotal Knowledge Base:

    - How to calculate pod consumption hours
    - How to calculate high watermark pod count

Please send any feedback you have to pks-feedback@pivotal.io.

Downloading Cluster Logs

To download cluster logs, perform the following steps:

1. Gather credential and IP address information for your BOSH Director, SSH into the Ops Manager VM, and use the BOSH CLI v2+ to log in to the BOSH Director from the Ops Manager VM. For more information, see Advanced Troubleshooting with the BOSH CLI.

2. After logging in to the BOSH Director, identify the name of your PKS deployment. For example:

   ```
   $ bosh -e pks deployments
   ```

   Your PKS deployment name begins with `pivotal-container-service` and includes a BOSH-generated hash.

3. Identify the names of the VMs you want to retrieve logs from by listing all VMs in your deployment. For example:

   ```
   $ bosh -e pks -d pivotal-container-service-aa1234567bc8de9f0a1c vms
   ```

4. Download the logs from the VM. For example:

   ```
   $ bosh -e pks \
   -d pivotal-container-service-aa1234567bc8de9f0a1c logs pks/0
   ```

   See the View Log Files section of the *Diagnostic Tools* topic for information about using cluster logs to diagnose issues in your PKS deployment.

Please send any feedback you have to pks-feedback@pivotal.io.

Monitoring PKS with Sinks

Page last updated:

This topic describes how to monitor Pivotal Container Service (PKS) deployments using sink resources.

Prerequisites

Using sink resources for monitoring requires that you have set up a log processing solution capable of log ingress over TCP as described in RFC 5424.

In addition, you must configure sinks in PKS to send logs to that destination. For information on how to create and manage sinks in PKS, see Creating Sink Resources.

Note: Sinks created in PKS only support TCP connections. UDP connections are not currently supported.

About Sink Log Entries

Sinks and ClusterSinks include both pod logs as well as events from the Kubernetes API.

These logs and events are combined in a shared format to provide operators with a robust set of filtering and monitoring options.

Sink data has the following characteristics. All entries:

- Are timestamped.
- Contain the host ID of the BOSH-defined VM.
- Are annotated with a set of structured data, which includes the namespace, the object name or pod ID, and the container name.

Sink Log Entry Format

All sink log entries use the following format:

```
APP-NAME/NAMESPACE/POD-ID/CONTAINER-NAME
```

Where:

- `APP-NAME` is `pod.log` or `k8s.event`.
- `NAMESPACE` is the namespace associated with the pod log or Kubernetes event.
- `POD-ID` is the ID of the pod associated with the pod log or Kubernetes event.
- `CONTAINER-NAME` is the deployment associated with the pod log or Kubernetes event.

Pod Logs

Pod logs entries are distinguished by the string `pod.log` in the `APP-NAME` field.

Pod Log Example

The following is a sample pod log entry:

```
36 <14>1 2018-11-26T18:51:41.647825+00:00 vm-3ebfe45d-492d-4bfd-59c4-c45d91688c65
pod.log/rocky-raccoon/logspewer-6b58b6689d-dhddj - - [kubernetes@47450
app="logspewer" pod-template-hash="2614622458" namespace_name="rocky-raccoon"
object_name="logspewer-6b58b6689d-dhddj" container_name="logspewer"]
2018/11/26 18:51:41 Log Message 589910
```

Where:

- `vm-3ebfe45d-492d-4bfd-59c4-c45d91688c65` is the host ID of the BOSH VM.
- `pod.log` is the `APP-NAME`.
- `rocky-raccoon` is the `NAMESPACE`.
- `logspewer-6b58b6689d-dhddj` is the `POD-ID`.

Kubernetes API Events

Kubernetes API Event entries are distinguished by the string `k8s.event` in the `APP-NAME` field.

Kubernetes API Event Example

The following is an example Kubernetes API event log entry:

```
Nov 14 16:01:49 vm-b409c60e-2517-47ac-7c5b-2cd302287c3a
k8s.event/rocky-raccoon/logspewer-6b58b6689d-j9n:
Successfully assigned rocky-raccoon/logspewer-6b58b6689d-j9nq7
to vm-38dfd896-bb21-43e4-67b0-9d2f339adaf1
```

Where:

- `vm-b409c60e-2517-47ac-7c5b-2cd302287c3a` the host ID of the BOSH VM.
- `k8s.event` is the `APP-NAME`.
- `rocky-raccoon` is the `NAMESPACE`.
- `logspewer-6b58b6689d-j9n` is the `POD-ID`.

Notable Kubernetes API Events

The following section lists Kubernetes API Events that can help assess any Kubernetes scheduling problems in PKS.

To monitor for these events, look for log entries that contain the **Identifying String** indicated below for each event.

Failure to Retrieve Containers from Registry

ImagePullBackOff	
Description	Image pull back offs occur when the Kubernetes API cannot reach a registry to retrieve a container or the container does not exist in the registry. The scheduler might be trying to access a registry that is not available on the network. For example, Docker Hub is blocked by a firewall. Other reasons might include the registry is experiencing an outage or a specified container has been deleted or was never uploaded.
Identifying String	`Error:ErrImagePull`
Example Sink Log Entry	`Jan 25 10:18:58 gke-bf-test-default-pool-aa8027bc-rnf6 k8s.event/default/test-669d4d66b9-zd9h4/: Error: ErrImagePull`

Malfunctioning Containers

CrashLoopBackOff	
Description	Crash loop back offs imply that the container is not functioning as intended. There are several potential causes of crash loop back offs which depend on the related workload. To investigate further, examine the logs for that workload.
Identifying String	`Back-off restarting failed container`

Example Sink Log Entry	`Jan 25 09:26:44 vm-bfdfedef-4a6a-4c36-49fc-8b290ad42623 k8s.event/monitoring/cost-analyzer-prometheus-se: Back-off restarting failed container`

Successful Scheduling of Containers

ContainerCreated	
Description	Operators can monitor the creation and successful start of containers to keep track of platform usage at a high level. Cluster users can track this event to monitor the usage of their cluster.
Identifying String	`Started container`

Example Sink Log Entries	`Jan 25 09:14:55 35.239.18.250 k8s.event/rocky-raccoon/logspewer-6b58b6689d/: Created pod: logspewer-6b58b6689d-sr96t`
	`Jan 25 09:14:55 35.239.18.250 k8s.event/rocky-raccoon/logspewer-6b58b6689d-sr9: Successfully assigned rocky-raccoon/ logspewer-6b58b6689d-sr96t to vm-efe48928-be8e-4db5-772c-426ee7aa52f2`
	`Jan 25 09:14:55 vm-efe48928-be8e-4db5-772c-426ee7aa52f2 k8s.event/rocky-raccoon/logspewer-6b58b6689d-mkd: Killing container with id docker://logspewer:Need to kill Pod`
	`Jan 25 09:14:56 vm-efe48928-be8e-4db5-772c-426ee7aa52f2 k8s.event/rocky-raccoon/logspewer-6b58b6689d-sr9: Container image "oratos/logspewer:v0.1" already present on machine`
	`Jan 25 09:14:56 vm-efe48928-be8e-4db5-772c-426ee7aa52f2 k8s.event/rocky-raccoon/logspewer-6b58b6689d-sr9: Created container`
	`Jan 25 09:14:56 vm-efe48928-be8e-4db5-772c-426ee7aa52f2 k8s.event/rocky-raccoon/logspewer-6b58b6689d-sr9: Started container`

Failure to Schedule Containers

FailedScheduling	
Description	This event occurs when a container cannot be scheduled. For instance, this may occur due to lack of node resources..
Identifying String	`Insufficient RESOURCE` where `RESOURCE` is a specific type of resource. For example, `cpu`.
Example Sink Log Entries	`Jan 25 10:51:48 gke-bf-test-default-pool-aa8027bc-rnf6 k8s.event/default/test2-5c87bf4b65-7fdtd/: 0/1 nodes are available: 1 Insufficient cpu.`

Related Links

For more information on sinks in PKS, see the following topics:

- For information about creating sinks in PKS, see Creating Sink Resources.
- For information about sink architecture, see Sink Architecture in PKS.

Please send any feedback you have to pks-feedback@pivotal.io.

Monitoring Master/etcd Node VMs

Page last updated:

This topic includes information about monitoring the master/etcd node VMs in your Pivotal Container Service (PKS) deployment. You can monitor Kubernetes cluster health by monitoring and gathering metrics from etcd.

PKS colocates etcd, an open source distributed key value store, on Kubernetes master node VMs. The master node VMs use etcd for service discovery and configuration sharing within the cluster.

For more information about etcd, see the etcd documentation on GitHub.

For more information about configuring master/etcd nodes in the PKS tile, see the Plans section of *Installing PKS* for your IaaS:

- vSphere
- vSphere with NSX-T Integration
- Google Cloud Platform (GCP)
- Amazon Web Services (AWS)

Monitor etcd

The etcd VM provides monitoring data on its client port. You can enable the `/debug` endpoint for more verbose logging, but this can decrease cluster performance.

For more information about monitoring etcd, see Monitoring etcd on GitHub.

Gather Metrics from etcd

Each etcd VM exposes metrics on a `/metrics` endpoint. Connect a metrics system to etcd to gather information from the endpoint about cluster health.

You can configure any monitoring system of your choice to gather metrics. For example, the etcd documentation recommends using the open source Prometheus monitoring service. For more information, see the Prometheus documentation.

Troubleshoot etcd

We recommend working with Pivotal or VMware Support to troubleshoot master/etcd node VMs. The monitoring and metrics data you gather from the master/etcd node VMs can help the Support team diagnose and troubleshoot errors.

Please send any feedback you have to pks-feedback@pivotal.io.

Backing up and Restoring Enterprise PKS

Page last updated:

This section describes how to back up and restore the Pivotal Container Service (PKS) control plane and PKS clusters. PKS uses the Cloud Foundry BOSH Backup and Restore framework to back up and restore the PKS control plane and clusters.

BBR backs up the following PKS control plane components:

- UAA MySQL database
- PKS API MySQL database

BBR backs up the following cluster components:

- etcd database

BBR orchestrates triggering the backup or restore process on the BOSH deployment, and transfers the backup artifacts to and from the BOSH deployment.

For more information about installing and using BBR, see the following topics:

- Installing BOSH Backup and Restore
- Backing up PKS
- Restore the BOSH Director.
- Restore the PKS Control Plane.
- Restore PKS Clusters.

For information about troubleshooting BBR, see BBR Logging.

Please send any feedback you have to pks-feedback@pivotal.io.

Installing BOSH Backup and Restore

Page last updated:

This topic describes how to install BOSH Backup and Restore (BBR).

Overview

To install BBR, first validate that your jumpbox VM is a valid BOSH backup host, then copy the BBR executable to the jumpbox.

After installing BBR, you can run `bbr` commands to back up and restore your PKS deployment.

For more information about using BOSH Backup and Restore, see:

- To perform a backup, see Backing up PKS.
- To perform a restore of the BOSH Director, see Restore the BOSH Director.
- To perform a restore of the PKS Control Plane, see Restore the PKS Control Plane.
- To perform a restore of the PKS Clusters. see Restore PKS Clusters.

Prerequisite

Using BBR requires the following:

- A jumpbox. You must have a jumpbox before you can install BBR to the jumpbox.
- A bbr executable file. You must have the correct BBR executable version for your PKS installation.

A jumpbox is a separate, hardened server on your network that provides a controlled means of accessing the other VMs on your network. See the jumpbox-deployment GitHub repository for an example jumpbox deployment.

To determine the correct version of BBR for your deployment, see the PKS Release Notes. To download a BBR installation file, see BOSH Backup and Restore on the Pivotal Network.

Step 1: Configure Your Jumpbox

Configure your jumpbox to meet the following requirements:

- Your jumpbox must be able to communicate with the network that contains your PKS deployment. You can use the Ops Manager VM as your jumpbox.
- Your jumpbox must have sufficient space for the backup.
- Your jumpbox must be in the same network as the deployed VMs because BBR connects to the VMs at their private IP addresses. BBR does not support SSH gateways.
- Your jumpbox should be a host with minimal network latency to the source VMs you are configuring BBR to backup.

Note: BBR uses SSH to orchestrate the backup of your PKS instances using port 22 by default.

Step 2: Transfer BBR to Your Jumpbox

Copy the `bbr` executable to a local disk then upload the executable to the jumpbox:

1. Download the latest BOSH Backup and Restore release from Pivotal Network.
2. To add executable permissions to the `bbr` binary file, run the following command:

```
chmod a+x bbr
```

3. To securely copy the `bbr` binary file to your jumpbox, run the following command:

```
scp LOCAL-PATH-TO-BBR/bbr JUMPBOX-USER@JUMPBOX-ADDRESS:
```

Where:

- `LOCAL-PATH-TO-BBR` is the path to the `bbr` binary you downloaded from Pivotal Network.
- `JUMPBOX-USER` is the ssh username for connecting to the jumpbox.
- `JUMPBOX-ADDRESS` is the IP address, or hostname, of the jumpbox.

Please send any feedback you have to pks-feedback@pivotal.io.

Backing Up PKS

Page last updated:

This topic describes how to use BOSH Backup and Restore (BBR) to back up the Pivotal Container Service (PKS) Control Plane and its cluster deployments.

Overview

The BOSH Director, PKS Control Plane, and cluster deployments include custom backup and restore scripts which encapsulate the correct procedure for backing up and restoring the Director and Control Plane.

BBR orchestrates running the backup and restore scripts and transferring the generated backup artifacts to and from a backup directory. If configured correctly, BBR can use TLS to communicate securely with backup targets.

- To perform a restore of the BOSH Director, see Restore the BOSH Director.
- To perform a restore of the PKS Control Plane, see Restore the PKS Control Plane.
- To perform a restore of a cluster deployment, see Restore PKS Clusters.

To view the BBR release notes, see the Cloud Foundry documentation, BOSH Backup and Restore Release Notes.

Supported Components

BBR can backup the following components:

- BOSH Director
- PKS control plane UAA MySQL database
- PKS control plane PKS API MySQL database
- The ETCD database of a cluster.

Unsupported Components

BBR can not be used to back up the following components:

- Harbor tile
- Persistent volumes attached to nodes.
- Network resources e.g. Load Balancers to the cluster

BBR has not been validated for backing up the following components:

- vSphere with NSX-T Objects

Preparing to Back Up

Before using BBR you must perform the following steps:

- Verify your BBR Version
- Download the BBR SSH Credentials
- Download the BOSH Director Credentials
- Download the UAA Client Credentials
- Retrieve the BOSH Director Address
- Download the Root CA Certificate
- Download the BOSH Command Line Credentials
- Retrieve your Cluster Deployment Name

Verify Your BBR Version

Before running BBR, you must verify that the installed version of BBR is compatible with your deployment's current PKS release.

1. For your current PKS release's minimum version information, see the PKS Release Notes.
2. To verify the currently installed BBR version, run the following command:

   ```
   bbr version
   ```

   If you do not have BBR installed, or your installed version does not meet the minimum version requirement, see Installing BOSH Backup and Restore.

Collect Credentials and Account Information

Before you can perform a backup you will need to collect accounts and credentials to authenticate into your jumpbox, BOSH Director, and Ops Manager.

Download the BBR SSH Credentials

There are two ways to retrieve BOSH Director credentials:

- Ops Manager Installation Dashboard
- Ops Manager API

Ops Manager Installation Dashboard

To retrieve your **Bbr Ssh Credentials** using the Ops Manager Installation Dashboard, perform the following steps:

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the BOSH Director tile.
3. Click the **Credentials** tab.
4. Locate **Bbr Ssh Credentials**.
5. Click **Link to Credentials** next to it.
6. Copy the `private_key_pem` field value.

Ops Manager API

To retrieve your **Bbr Ssh Credentials** using the Ops Manager API, perform the following steps:

1. Obtain your UAA access token. For more information, see Access the API
2. Retrieve the **Bbr Ssh Credentials** by running the following command:

   ```
   curl "https://OPS-MAN-FQDN/api/v0/deployed/director/credentials/bbr_ssh_credentials" \
   -X GET \
   -H "Authorization: Bearer UAA-ACCESS-TOKEN"
   ```

   Where:

   - `OPS-MAN-FQDN` is the fully-qualified domain name (FQDN) for your Ops Manager deployment.
   - `UAA-ACCESS-TOKEN` is your UAA access token.

3. Copy the value of the `private_key_pem` field.

Save the BBR SSH Credentials to File

1. To reformat the copied `private_key_pem` value and save it to a file in the current directory, run the following command:

   ```
   printf -- "YOUR-PRIVATE-KEY" > PRIVATE-KEY-FILE
   ```

   Where:

   - `YOUR-PRIVATE-KEY` is the text of your private key.
   - `PRIVATE-KEY-FILE` is the path to the private key file you are creating.

   For example:

   ```
   $ printf --  "-----begin rsa private key----- fake key contents ----end rsa private key-----" > bbr_key.pem
   ```

Download the BOSH Director Credentials

There are two ways to retrieve BOSH Director credentials:

- Ops Manager Installation Dashboard
- Ops Manager API

Ops Manager Installation Dashboard

To retrieve your BOSH Director credentials using the Ops Manager Installation Dashboard, perform the following steps:

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the BOSH Director tile.
3. Click the **Credentials** tab.
4. Locate **Director Credentials**.
5. Click **Link to Credentials** next to it.
6. Copy the value of the `private_key_pem` field.

Ops Manager API

To retrieve your BOSH Director credentials using the Ops Manager API, perform the following steps:

1. Obtain your UAA access token. For more information, see Access the Ops Manager API
2. Retrieve the **Director Credentials** by running the following command:

   ```
   curl "https://OPS-MAN-FQDN/api/v0/deployed/director/credentials/bbr_ssh_credentials" \
   -X GET \
   -H "Authorization: Bearer UAA-ACCESS-TOKEN"
   ```

   Where: `OPS-MAN-FQDN` is the fully-qualified domain name (FQDN) for your Ops Manager deployment. `UAA-ACCESS-TOKEN` is your UAA access token.

3. Copy the value of the `private_key_pem` field.

Save the BOSH Director Credentials to a File

1. To reformat the copied `private_key_pem` value and save it a file in the current directory, run the following command:

   ```
   printf -- "YOUR-PRIVATE-KEY" > PRIVATE-KEY-FILE
   ```

   Where:

- `YOUR-PRIVATE-KEY` is the text of your private key.
- `PRIVATE-KEY-FILE` is the path to the private key file you are creating.

For example:

```
$ printf --  "-----begin rsa private key----- fake key contents ----end rsa private key-----" > bbr_key.pem
```

Download the UAA Client Credentials

To obtain BOSH credentials for your BBR operations, perform the following steps:

1. From the Ops Manager Installation Dashboard, click the **Pivotal Container Service** tile.
2. Select the **Credentials** tab.
3. Navigate to **Credentials > UAA Client Credentials**.
4. Record the value for `uaa_client_secret`.
5. Record the value for `uaa_client_name`.

Note: You must use BOSH credentials that limit the scope of BBR activity to your cluster deployments.

Retrieve the BOSH Director Address

You access the BOSH Director using an IP address.

To obtain your BOSH Director's IP address:

1. Open the Ops Manager Installation Dashboard.
2. Select **BOSH Director > Status**.
3. Select the listed Director IP Address.

Log In to BOSH Director

1. If you are not using the Ops Manager VM as your jumpbox, install the latest BOSH CLI on your jumpbox.
2. To log in to BOSH Director, using the IP address that you recorded above, run the following command line:

   ```
   bosh -e BOSH-DIRECTOR-IP \
   --ca-cert PATH-TO-BOSH-SERVER-CERTIFICATE log-in
   ```

   Where:

   - `BOSH-DIRECTOR-IP` is the BOSH Director IP address recorded above.
   - `PATH-TO-BOSH-SERVER-CERTIFICATE` is the path to the root Certificate Authority (CA) certificate as outlined in Download the Root CA Certificate.
3. To specify **Email**, specify `director`.
4. To specify **Password**, enter the **Director Credentials** that you obtained in Download the BOSH Director Credentials.
   For example:

   ```
   $ bosh -e 10.0.0.3 \
   --ca-cert /var/tempest/workspaces/default/root_ca_certificate log-in
   Email (): director
   Password (): ******************
   Successfully authenticated with UAA
   Succeeded
   ```

Download the Root CA Certificate

To download the root CA certificate for your PKS deployment, perform the following steps:

1. Open the Ops Manager Installation Dashboard
2. In the top right corner, click your username.
3. Navigate to **Settings** > **Advanced**.
4. Click **Download Root CA Cert**.

Download the BOSH Command Line Credentials

1. Open the Ops Manager Installation Dashboard.
2. Click the **BOSH Director** tile.
3. In the BOSH Director tile, click the **Credentials** tab.
4. Navigate to **Bosh Commandline Credentials**.
5. Click **Link to Credential**.
6. Copy the credential value.

Retrieve Your Cluster Deployment Name

To locate and record your cluster's BOSH deployment name, follow the steps below.

1. On the command line, run the following command to log in:

   ```
   pks login -a PKS-API -u USERNAME -k
   ```

   Where:

   - `PKS-API` is the domain name for the PKS API that you entered in **Ops Manager** > **Pivotal Container Service** > **PKS API** > **API Hostname (FQDN)**. For example, `api.pks.example.com`.
   - `USERNAME` is your user name.

     See Logging in to PKS for more information about the `pks login` command.

2. To find the cluster ID associated with the cluster you want to back up, run the following command:

   ```
   pks cluster CLUSTER-NAME
   ```

   Where `CLUSTER-NAME` is the name of your cluster.

3. From the output of this command, record the **UUID** value.
4. Open the Ops Manager Installation Dashboard
5. Click the **BOSH Director** tile.
6. Select the **Credentials** tab.
7. Navigate to **Bosh Commandline Credentials** and click **Link to Credential**.
8. Copy the credential value.
9. SSH into your jumpbox. For more information about the jumpbox, see Installing BOSH Backup and Restore.
10. To retrieve your cluster BOSH deployment name, run the following command:

    ```
    BOSH-CLI-CREDENTIALS deployments | grep UUID
    ```

Where:

- `BOSH-CLI-CREDENTIALS` is the full value that you copied from the BOSH Director tile in Download the BOSH Command Line Credentials.
- `UUID` is the cluster UUID that you recorded in the previous step.

Back Up PKS

To back up your PKS environment you must first connect to your jumpbox before executing `bbr` backup commands.

Connect to Your Jumpbox

You can establish a connection to your jumpbox in one of the following ways:

- Connect with SSH
- Connect with BOSH_ALL_PROXY

For general information about the jumpbox, see Installing BOSH Backup and Restore.

Connect with SSH

To connect to your jumpbox with SSH, do one of the following:

- **If you are using the Ops Manager VM as your jumpbox, log in to the Ops Manager VM.** See Log in to the Ops Manager VM with SSH in *Advanced Troubleshooting with the BOSH CLI*.

- **If you want to connect to your jumpbox using the command line, run the following command:**

  ```
  ssh -i PATH-TO-KEY USER@IP-ADDRESS
  ```

  Where:

  - `PATH-TO-KEY` is the local path to your private key for the jumpbox host.
  - `USER` is your jumpbox username.
  - `IP-ADDRESS` is the IP address of your jumpbox.

Note: If you connect to your jumpbox with SSH, you must run the BBR commands in the following sections from within your jumpbox.

Connect with BOSH_ALL_PROXY

You can use the `BOSH_ALL_PROXY` environment variable to open an SSH tunnel with SOCKS5 to your jumpbox. This tunnel enables you to forward requests from your local machine to the BOSH Director through the jumpbox. When `BOSH_ALL_PROXY` is set, BBR always uses its value to forward requests to the BOSH Director.

Note: For the following procedures to work, ensure the SOCKS port is not already in use by a different tunnel or process.

To connect with `BOSH_ALL_PROXY`, do one of the following:

- **If you want to establish the tunnel separate from the BOSH CLI, do the following:**

  1. Establish the tunnel and make it available on a local port by running the following command:

     ```
     ssh -4 -D SOCKS-PORT -fNC JUMPBOX-NAME@IP-ADDRESS -i JUMPBOX-KEY-FILE -o ServerAliveInterval=60
     ```

     Where:

     - `SOCKS-PORT` is your local SOCKS port.
     - `JUMPBOX-NAME` is the name of your jumpbox.

- `IP-ADDRESS` is the IP address of your jumpbox.
- `JUMPBOX-KEY-FILE` is your local SSH private key for accessing your jumpbox.

For example:

```
$ ssh -4 -D 12345 -fNC jumpbox@203.0.113.0 -i jumpbox.key -o ServerAliveInterval=60
```

2. Provide the BOSH CLI with access to the tunnel through `BOSH_ALL_PROXY` by running the following command:

```
export BOSH_ALL_PROXY=socks5://localhost:SOCKS-PORT
```

Where is `SOCKS-PORT` is your local SOCKS port.

- **If you want to establish the tunnel using the BOSH CLI, do the following:**

1. Provide the BOSH CLI with the necessary SSH credentials to create the tunnel by running the following command:

```
export BOSH_ALL_PROXY=ssh+socks5://JUMPBOX-NAME@IP-ADDRESS:SOCKS-PORT?private_key=JUMPBOX-KEY-FILE
```

Where:

- `JUMPBOX-NAME` is the name of your jumpbox.
- `IP-ADDRESS` is the IP address of your jumpbox.
- `SOCKS-PORT` is your local SOCKS port.
- `JUMPBOX-KEY-FILE` is the local SSH private key for accessing the jumpbox.

For example:

```
$ export BOSH_ALL_PROXY=ssh+socks5://jumpbox@203.0.113.0:12345?private_key=jumpbox.key
```

Note: Using `BOSH_ALL_PROXY` can result in longer backup and restore times because of network performance degradation. All operations must pass through the proxy which means moving backup artifacts can be significantly slower.

Warning: In BBR v1.5.0 and earlier, the tunnel created by the BOSH CLI does not include the `ServerAliveInterval` flag. This may result in your SSH connection timing out when transferring large artifacts. In BBR v1.5.1, the `ServerAliveInterval` flag is included. For more information, see bosh-backup-and-restore v1.5.1 on GitHub.

Back Up Installation Settings

To ensure your BBR backup is reliable, your should also frequently export your Ops Manager installation settings as a backup.

There are two ways to export Ops Manager installation settings:

- Export settings using the Ops Manager UI
- Export settings using the Ops Manager API

Note: If you want to automate the back up process, you can use the Ops Manager API to export your installation settings.

When exporting your installation settings, keep in mind the following:

- You should always export your installation settings before following the steps in the Restore the BOSH Director section of the *Restoring PKS* topic.
- You can only export Ops Manager installation settings after you have deployed at least once.
- Your Ops Manager settings export is only a backup of Ops Manager configuration settings. The export is not a backup of your VMs or any external MySQL databases.
- Your Ops Manager settings export is encrypted. Make sure you keep track of your Decryption Passphrase because this is needed to restore the Ops Manager settings.

Export Settings Using the Ops Manager UI

To export your Ops Manager installation settings using the Ops Manager UI, perform the following steps:

1. From the **Installation Dashboard** in the Ops Manager interface, click your username at the top right navigation.
2. Select **Settings**.
3. Select **Export Installation Settings**.
4. Click **Export Installation Settings**.

Export Settings Using the Ops Manager API

To export your Ops Manager installation settings using the Ops Manager API, perform the following steps:

1. To export your installation settings using the Ops Manager API, run the following command:

```
curl https://OPS-MAN-FQDN/api/v0/installation_asset_collection \
-H "Authorization: Bearer UAA-ACCESS-TOKEN" > installation.zip
```

   Where:

   - `OPS-MAN-FQDN` is the fully-qualified domain name (FQDN) for your Ops Manager deployment.
   - `UAA-ACCESS-TOKEN` is your UAA access token. For more information, see Access the API.

Back Up the PKS BOSH Director

To back up BOSH Director you will validate your current configuration, then execute the `bbr` backup command.

Validate the PKS BOSH Director

1. To confirm that your BOSH Director is reachable and has the correct BBR scripts, run the following command:

```
bbr director  --host BOSH-DIRECTOR-IP  --username bbr \
--private-key-path PRIVATE-KEY-FILE  pre-backup-check
```

   Where:

   - `BOSH-DIRECTOR-IP` is the address of the BOSH Director. If the BOSH Director is public, `BOSH-DIRECTOR-IP` is a URL, such as `https://my-bosh.xxx.cf-app.com`. Otherwise, this is the internal IP `BOSH-DIRECTOR-IP` which you can retrieve as show in Retrieve the BOSH Director Address.
   - `PRIVATE-KEY-FILE` is the path to the private key file that you can create from `Bbr Ssh Credentials` as shown in Download the BBR SSH Credentials.

   For example:

```
$ bbr director --host 10.0.0.5 --username bbr \
--private-key-path private-key.pem pre-backup-check
```

2. If the pre-backup check command fails, perform the following actions:

   a. Run the command again, adding the `--debug` flag to enable debug logs. For more information, see BBR Logging.
   b. Make any correction suggested in the output and run the pre-backup check again.

Back Up the PKS BOSH Director

1. If the pre-backup check succeeds, run the BBR backup command from your jumpbox to back up the PKS BOSH Director:

```
bbr director --host BOSH-DIRECTOR-IP --username bbr \
--private-key-path PRIVATE-KEY-FILE  backup
```

   Where:

   - `BOSH-DIRECTOR-IP` is the address of the BOSH Director. If the BOSH Director is public, `BOSH-DIRECTOR-IP` is a URL, such as `https://my-bosh.xxx.cf-app.com`. Otherwise, this is the internal IP. See Retrieve the BOSH Director Address for more information.

- `PRIVATE-KEY-FILE` is the path to the private key file that you can create from `Bbr Ssh Credentials` as shown in Download the BBR SSH Credentials.

For example:

```
$ bbr director  --host 10.0.0.5 --username bbr \
--private-key-path private-key.pem  backup
```

Note: The BBR backup command can take a long time to complete. You can run it independently of the SSH session so that the process can continue running even if your connection to the jumpbox fails. The command above uses `nohup`, but you can run the command in a `screen` or `tmux` session instead.

2. If the command completes successfully, follow the steps in Manage Your Backup Artifact below.
3. If the backup command fails, perform the following actions:
   - Run the command again, adding the `--debug` flag to enable debug logs. For more information, see BBR Logging.
   - Follow the steps in Recover from a Failing Command.

Back Up the PKS Control Plane

To back up your PKS Control Plane you will validate the Control Plane, then execute the `bbr` backup command.

Locate the PKS Deployment Name

Locate and record your PKS BOSH deployment name as follows:

1. Open an SSH connection to either your jumpbox, as described in the previous section, or the Ops Manager VM. For instructions on how to SSH into the Ops Manager VM, see Log in to the Ops Manager VM with SSH in *Advanced Troubleshooting with the BOSH CLI*.
2. On the command line, run the following command to retrieve your PKS BOSH deployment name.

   ```
   BOSH-CLI-CREDENTIALS deployments | grep pivotal-container-service
   ```

   Where `BOSH-CLI-CREDENTIALS` is the full value that you copied from the BOSH Director tile in Download the BOSH Commandline Credentials.

   For example:

   ```
   $ BOSH_CLIENT=ops_manager BOSH_CLIENT_SECRET=p455w0rd BOSH_CA_CERT=/var/tempest/workspaces/default/root_ca_certificate BOSH_ENVIRONMENT=10.0.0.5 bosh

   pivotal-container-service-51f08f6402aaa960f041          backup-and-restore-sdk/1.8.0   bosh-google-kvm-ubuntu-xenial-go_agent/250.25
   service-instance_4ffeb5b5-5182-4faa-9d92-696d97cc9ae1   bosh-dns/1.10.0                bosh-google-kvm-ubuntu-xenial-go_agent/250.25
   pivotal-container-service-51f08f6402aaa960f041
   ```

3. Review the returned output. The PKS BOSH deployment name begins with `pivotal-container-service` and includes a unique identifier. In the example output above, the BOSH deployment name is `pivotal-container-service-51f08f6402aaa960f041`.

Validate the PKS Control Plane

1. To confirm that your PKS control plane is reachable and has a deployment that can be backed up, run the BBR pre-backup check command:

   ```
   BOSH_CLIENT_SECRET=BOSH-CLIENT-SECRET bbr deployment \
   --target BOSH-TARGET  --username BOSH-CLIENT --deployment DEPLOYMENT-NAME \
   --ca-cert PATH-TO-BOSH-SERVER-CERT \
   pre-backup-check
   ```

   Where:

   - `BOSH-CLIENT-SECRET` is your BOSH client secret. If you do not know your BOSH Client Secret, open your BOSH Director tile, navigate to **Credentials > Bosh Commandline Credentials** and record the value for `BOSH_CLIENT_SECRET`.
   - `BOSH-TARGET` is your BOSH Environment setting. If you do not know your BOSH Environment setting, open your BOSH Director tile, navigate to **Credentials > Bosh Commandline Credentials** and record the value for `BOSH_ENVIRONMENT`. You must be able to reach the target address from the workstation where you run `bbr` commands.

- `BOSH-CLIENT` is your BOSH Client Name. If you do not know your BOSH Client Name, open your BOSH Director tile, navigate to **Credentials > Bosh Commandline Credentials** and record the value for `BOSH_CLIENT`.
- `DEPLOYMENT-NAME` is the PKS BOSH deployment name that you located in the Locate the PKS Deployment Name section above.
- `PATH-TO-BOSH-CA-CERT` is the path to the root CA certificate that you downloaded in Download the Root CA Certificate above.

For example:

```
$ BOSH_CLIENT_SECRET=p455w0rd bbr deployment \
--target bosh.example.com --username admin --deployment cf-acceptance-0 \
--ca-cert bosh.ca.cert \
pre-backup-check
```

2. If the pre-backup check command fails, perform the following actions:

   a. Run the command again, adding the `--debug` flag to enable debug logs. For more information, see BBR Logging.
   b. Make any correction suggested in the output and run the pre-backup check again. For example, the deployment that you selected might not have the correct backup scripts, or the connection to the BOSH Director failed.

Back Up the PKS Control Plane

If the pre-backup check succeeds, run the BBR backup command.

1. To back up the PKS control plane, run the following BBR backup command from your jumpbox:

```
BOSH_CLIENT_SECRET=BOSH-CLIENT-SECRET nohup bbr deployment \
--target BOSH-TARGET --username BOSH-CLIENT --deployment DEPLOYMENT-NAME \
--ca-cert PATH-TO-BOSH-SERVER-CERT \
backup --with-manifest [--artifact-path]
```

   Where:

   - `BOSH-CLIENT-SECRET` is your BOSH client secret. If you do not know your BOSH Client Secret, open your BOSH Director tile, navigate to **Credentials > Bosh Commandline Credentials** and record the value for `BOSH_CLIENT_SECRET`.
   - `BOSH-TARGET` is your BOSH Environment setting. If you do not know your BOSH Environment setting, open your BOSH Director tile, navigate to **Credentials > Bosh Commandline Credentials** and record the value for `BOSH_ENVIRONMENT`. You must be able to reach the target address from the workstation where you run `bbr` commands.
   - `BOSH-CLIENT` is your BOSH Client Name. If you do not know your BOSH Client Name, open your BOSH Director tile, navigate to **Credentials > Bosh Commandline Credentials** and record the value for `BOSH_CLIENT`.
   - `DEPLOYMENT-NAME` is the PKS BOSH deployment name that you located in the Locate the PKS Deployment Name section above.
   - `PATH-TO-BOSH-CA-CERT` is the path to the root CA certificate that you downloaded in Download the Root CA Certificate above.
   - `--with-manifest` is necessary in order to redeploy your PKS Control Plane in the case of its loss. `--with-manifest` is an optional `backup` parameter to include the manifest in the backup artifact.
   - `--artifact-path` is an optional `backup` parameter to specify the output path for the backup artifact.

> **Note**: The `--with-manifest` flag is necessary in order to redeploy your PKS Control Plane in the case of its loss. The backup artifact created by this process contains credentials that you should keep secret.

   For example:

```
$ BOSH_CLIENT_SECRET=p455w0rd nohup bbr deployment \
--target bosh.example.com  --username admin --deployment cf-acceptance-0 \
--ca-cert bosh.ca.cert \
backup --with-manifest
```

> **Note**: The BBR backup command can take a long time to complete. You can run it independently of the SSH session so that the process can continue running even if your connection to the jumpbox fails. The command above uses `nohup`, but you can run the command in a `screen` or `tmux` session instead.

2. If the command completes successfully, follow the steps in Manage Your Backup Artifact below.

3. If the backup command fails, perform the following actions:

   a. Run the command again, adding the `--debug` flag to enable debug logs. For more information, see BBR Logging.
   b. Follow the steps in Recover from a Failing Command.

Back Up Cluster Deployments

Before backing up your PKS cluster deployments you should verify that they can be backed up.

Verify Your Cluster Deployments

To verify that you can reach your PKS cluster deployments and that the deployments can be backed up, follow the steps below.

1. SSH into your jumpbox. For more information about the jumpbox, see Configure Your Jumpbox in *Installing BOSH Backup and Restore*.

2. To perform the BBR pre-backup check, run the following command from your jumpbox:

   ```
   BOSH_CLIENT_SECRET=PKS-UAA-CLIENT-SECRET  bbr deployment \
   --all-deployments  --target BOSH-TARGET  --username PKS-UAA-CLIENT-NAME \
   --ca-cert PATH-TO-BOSH-SERVER-CERT \
   pre-backup-check
   ```

   Where:

   - `PKS-UAA-CLIENT-SECRET` is the value you recorded for `uaa_client_secret` in Download the UAA Client Credentials above.
   - `BOSH-TARGET` is the value you recorded for the BOSH Director's address in Retrieve the BOSH Director Address above. You must be able to reach the target address from the machine where you run `bbr` commands.
   - `PKS-UAA-CLIENT-NAME` is the value you recorded for `uaa_client_name` in Download the UAA Client Credentials above.
   - `PATH-TO-BOSH-SERVER-CERT` is the path to the root CA certificate that you downloaded in Download or Locate Root CA Certificate above.

   For example:

   ```
   $ BOSH_CLIENT_SECRET=p455w0rd  bbr deployment \
   --all-deployments --target bosh.example.com --username pivotal-container-service-12345abcdefghijklmn \
   --ca-cert /var/tempest/workspaces/default/root_ca_certificate \
   pre-backup-check
   ```

3. If the pre-backup-check command is successful, the command returns a list of cluster deployments that can be backed up.

   For example:

   ```
   [21:51:23] Pending: service-instance_abcdeg-1234-5678-hijk-90101112131415
   [21:51:23] ----------------------------
   [21:51:31] Deployment 'service-instance_abcdeg-1234-5678-hijk-90101112131415' can be backed up.
   [21:51:31] ----------------------------
   [21:51:31] Successfully can be backed up: service-instance_abcdeg-1234-5678-hijk-90101112131415
   ```

   In the output above, `service-instance_abcdeg-1234-5678-hijk-90101112131415` is the BOSH deployment name of a PKS cluster.

4. If the pre-backup-check command fails, do one or more of the following:

   - Make sure you are using the correct Pivotal Container Service credentials.
   - Run the command again, adding the `--debug` flag to enable debug logs. For more information, see BBR Logging.
   - Make the changes suggested in the output and run the pre-backup check again. For example, the deployments might not have the correct backup scripts, or the connection to the BOSH Director failed.

Back Up Cluster Deployments

When backing up your PKS cluster, you can choose to back up only one cluster or to backup all cluster deployments in scope. The procedures to do this are the following:

- Back up All Cluster Deployments
- Back Up One Cluster Deployment

Back Up All Cluster Deployments

The following procedure backs up all cluster deployments.

Make sure you use the PKS UAA credentials that you recorded in Download the UAA Client Credentials. These credentials limit the scope of the backup to cluster deployments only.

Note: The BBR backup command can take a long time to complete. You can run it independently of the SSH session so that the process can continue running even if your connection to the jumpbox fails. The command above uses `nohup`, but you could also run the command in a `screen` or `tmux` session.

1. To back up all cluster deployments, run the following command from your jumpbox:

```
BOSH_CLIENT_SECRET=PKS-UAA-CLIENT-SECRET nohup bbr deployment \
--all-deployments --target BOSH-TARGET --username PKS-UAA-CLIENT-NAME \
--ca-cert PATH-TO-BOSH-SERVER-CERT \
backup [--with-manifest] [--artifact-path]
```

   Where:

   - `PKS-UAA-CLIENT-SECRET` is the value you recorded for `uaa_client_secret` in Download the UAA Client Credentials above.
   - `BOSH-TARGET` is the value you recorded for the BOSH Director's address in Retrieve the BOSH Director Address above. You must be able to reach the target address from the machine where you run `bbr` commands.
   - `PKS-UAA-CLIENT-NAME` is the value you recorded for `uaa_client_name` in Download the UAA Client Credentials above.
   - `PATH-TO-BOSH-SERVER-CERT` is the path to the root CA certificate that you downloaded in Download the Root CA Certificate above.
   - `--with-manifest` is an optional `backup` parameter to include the manifest in the backup artifact. If you use this flag, the backup artifact then contains credentials that you should keep secret.
   - `--artifact-path` is an optional `backup` parameter to specify the output path for the backup artifact.

   For example:

```
$ BOSH_CLIENT_SECRET=p455w0rd \
nohup bbr deployment \
--all-deployments \
--target bosh.example.com \
--username pivotal-container-service-12345abcdefghijklmn \
--ca-cert /var/tempest/workspaces/default/root_ca_certificate \
backup
```

   Note: The optional `--with-manifest` flag directs BBR to create a backup containing credentials. You should manage the generated backup artifact knowing it contains secrets for administering your environment.

2. If the `backup` command completes successfully, follow the steps in Manage Your Backup Artifact below.

3. If the backup command fails, the backup operation exits. BBR does not attempt to continue backing up any non-backed up clusters. To troubleshoot a failing backup, do one or more of the following:

   - Run the command again, adding the `--debug` flag to enable debug logs. For more information, see BBR Logging.
   - Follow the steps in Recover from a Failing Command below.

Back Up One Cluster Deployment

1. To backup a single, specific cluster deployment, run the following command from your jumpbox:

```
BOSH_CLIENT_SECRET=PKS-UAA-CLIENT-SECRET \
nohup bbr deployment \
--deployment CLUSTER-DEPLOYMENT-NAME \
--target BOSH-DIRECTOR-IP \
--username PKS-UAA-CLIENT-NAME \
--ca-cert PATH-TO-BOSH-SERVER-CERT \
backup [--with-manifest] [--artifact-path]
```

   Where:

   - `PKS-UAA-CLIENT-SECRET` is the value you recorded for `uaa_client_secret` in Download the UAA Client Credentials above.
   - `CLUSTER-DEPLOYMENT-NAME` is the value you recorded in Retrieve your Cluster Deployment Name above.
   - `BOSH-TARGET` is the value you recorded for the BOSH Director's address in Retrieve the BOSH Director Address above. You must be able to reach the target address from the machine where you run `bbr` commands.
   - `PKS-UAA-CLIENT-NAME` is the value you recorded for `uaa_client_name` in Download the UAA Client Credentials above.
   - `PATH-TO-BOSH-SERVER-CERT` is the path to the root CA certificate that you downloaded in Download the Root CA Certificate above.
   - `--with-manifest` is an optional `backup` parameter to include the manifest in the backup artifact. If you use this flag, the backup artifact

then contains credentials that you should keep secret.
- `--artifact-path` is an optional `backup` parameter to specify the output path for the backup artifact.

For example:

```
$ BOSH_CLIENT_SECRET=p455w0rd  nohup bbr deployment \
--deployment service-instance_abcdeg-1234-5678-hijk-9010111213141 \
--target bosh.example.com  --username pivotal-container-service-12345abcdefghijklmn \
--ca-cert /var/tempest/workspaces/default/root_ca_certificate \
backup
```

Note: The optional `--with-manifest` flag directs BBR to create a backup containing credentials. You should manage the generated backup artifact knowing it contains secrets for administering your environment.

2. If the `backup` command completes successfully, follow the steps in Manage Your Backup Artifact below.
3. If the backup command fails, do one or more of the following:
   - Run the command again, adding the `--debug` flag to enable debug logs. For more information, see BBR Logging.
   - Follow the steps in Recover from a Failing Command below.

Cancel a Backup

Backups can take a long time. If you realize that the backup is going to fail or that your developers need to push an app immediately, you might need to cancel the backup.

To cancel a backup, perform the following steps:

1. Terminate the BBR process by pressing Ctrl-C and typing `yes` to confirm.
2. Because stopping a backup can leave the system in an unusable state and prevent additional backups, follow the procedures in Clean up After a Failed Backup below.

After Backing Up

After the backup has completed you should review and manage the generated backup artifacts.

Manage Your Backup Artifact

The BBR-created backup consists of a directory containing the backup artifacts and metadata files. BBR stores each completed backup directory within the current working directory.

Note: The optional `--with-manifest` flag directs BBR to create a backup containing credentials. You should manage the generated backup artifact knowing it contains secrets for administering your environment.

BBR backup artifact directories are named using the following formats:

- `DIRECTOR-IP-TIMESTAMP` for the BOSH Director backups.
- `DEPLOYMENT-TIMESTAMP` for the Control Plane backup.
- `DEPLOYMENT-TIMESTAMP` for the cluster deployment backups.

Keep your backup artifacts safe by following these steps:

1. Move the backup artifacts off the jumpbox to your storage space.
2. Compress and encrypt the backup artifacts when storing them.
3. Make redundant copies of your backup and store them in multiple locations. This minimizes the risk of losing your backups in the event of a disaster.
4. Each time you redeploy PKS, test your backup artifact by following the procedures in:

- Restore the PKS BOSH Director.
- Restore the PKS Control Plane.
- Restore PKS Clusters.

Recover From a Failing Command

If the backup fails, follow these steps:

1. Ensure that you set all the parameters in the backup command.
2. Ensure the credentials previously obtained are valid.
3. Ensure the deployment that you specify in the BBR command exists.
4. Ensure that the jumpbox can reach the BOSH Director.
5. Consult BBR Logging.
6. If you see the error message: `Directory /var/vcap/store/bbr-backup already exists on instance`, run the appropriate cleanup command. See Clean Up After a Failed Backup below for more information.
7. If the backup artifact is corrupted, discard the failing artifacts and run the backup again.

Clean Up After a Failed Backup

If your backup process fails, use the BBR cleanup script to clean up the failed run.

⚠ **Warning**: It is important to run the BBR cleanup script after a failed BBR backup run. A failed backup run might leave the BBR backup directory on the instance, causing any subsequent attempts to backup to fail. In addition, BBR might not have run the post-backup scripts, leaving the instance in a locked state.

- If the PKS BOSH Director backup failed, run the following BBR cleanup script command to clean up:

```
bbr director --host BOSH-DIRECTOR-IP \
--username bbr  --private-key-path PRIVATE-KEY-FILE \
backup-cleanup
```

Where:

- `BOSH-DIRECTOR-IP` is the address of the BOSH Director. If the BOSH Director is public, `BOSH-DIRECTOR-IP` is a URL, such as `https://my-bosh.xxx.cf-app.com`. Otherwise, this is the internal IP `BOSH-DIRECTOR-IP` which you can retrieve as show in Retrieve the BOSH Director Address above.
- `PRIVATE-KEY-FILE` is the path to the private key file that you can create from `Bbr Ssh Credentials` as shown in Download the BBR SSH Credentials above. Replace the placeholder text using the information in the following table.

For example:

```
$ bbr director  --host 10.0.0.5  --username bbr \
--private-key-path private-key.pem \
backup-cleanup
```

- If the PKS control plane or PKS clusters backups fail, run the following BBR cleanup script command to clean up:

```
BOSH_CLIENT_SECRET=BOSH-CLIENT-SECRET \
bbr deployment \
--target BOSH-TARGET \
--username BOSH-CLIENT \
--deployment DEPLOYMENT-NAME \
--ca-cert PATH-TO-BOSH-CA-CERT \
backup-cleanup
```

Where:

- `BOSH-CLIENT-SECRET` is your BOSH client secret. If you do not know your BOSH Client Secret, open your BOSH Director tile, navigate to **Credentials > Bosh Commandline Credentials** and record the value for `BOSH_CLIENT_SECRET`.

- `BOSH-TARGET` is your BOSH Environment setting. If you do not know your BOSH Environment setting, open your BOSH Director tile, navigate to **Credentials > Bosh Commandline Credentials** and record the value for `BOSH_ENVIRONMENT`. You must be able to reach the target address from the workstation where you run `bbr` commands.
- `BOSH-CLIENT` is your BOSH Client Name. If you do not know your BOSH Client Name, open your BOSH Director tile, navigate to **Credentials > Bosh Commandline Credentials** and record the value for `BOSH_CLIENT`.
- `DEPLOYMENT-NAME` is the PKS BOSH deployment name that you located in the Locate the PKS Deployment Names section above.
- `PATH-TO-BOSH-CA-CERT` is the path to the root CA certificate that you downloaded in Download the Root CA Certificate above.

For example:

```
$ BOSH_CLIENT_SECRET=p455w0rd bbr deployment \
--target bosh.example.com --username admin --deployment cf-acceptance-0 \
--ca-cert bosh.ca.crt \
backup-cleanup
```

If the cleanup script fails, consult the following table to match the exit codes to an error message.

Value	Error
0	Success
1	General failure
8	The post-backup unlock failed. One of your deployments might be in a bad state and require attention.
16	The cleanup failed. This is a non-fatal error indicating that the utility has been unable to clean up open BOSH SSH connections to a deployment's VMs. Manual cleanup might be required to clear any hanging BOSH users and connections.

Please send any feedback you have to pks-feedback@pivotal.io.

Restoring PKS

Page last updated:

This topic describes how to use BOSH Backup and Restore (BBR) to restore the BOSH Director, and the Pivotal Container Service (PKS) Control Plane and Clusters.

Overview

In the event of a disaster, you may lose your environment's VMs, disks, and your IaaS network and load balancer resources as well. You can re-create your environment, configured with your saved PKS Ops Manager Installation settings, using your BBR backup artifacts.

Before restoring using BBR:

- Review the requirements listed in Compatibility of Restore.
- Complete all of the steps documented in Preparing to Restore a Backup.

Use BBR to restore the following:

- The BOSH Director plane, see Restore the BOSH Director.
- The PKS control plane, see Restore PKS Control Plane.
- The PKS Clusters, see Restore PKS Clusters.

Compatibility of Restore

The following are the requirements for a backup artifact to be restorable to another environment:

- **Topology**: BBR requires the BOSH topology of a deployment to be the same in the restore environment as it was in the backup environment.
- **Naming of instance groups and jobs**: For any deployment that implements the backup and restore scripts, the instance groups and jobs must have the same names.
- **Number of instance groups and jobs**: For instance groups and jobs that have backup and restore scripts, the same number of instances must exist.

Additional considerations:

- **Limited validation**: BBR puts the backed up data into the corresponding instance groups and jobs in the restored environment, but cannot validate the restore beyond that.
- **Same Cluster**: Currently, BBR supports the in-place restore of a cluster backup artifact onto the same cluster. Migration from one cluster to another using a BBR backup artifact has not yet been validated.

Note: This section is for guidance only. You should always validate your backups by using the backup artifacts in a restore.

Preparing to Restore a Backup

The steps for preparing to restore a BBR backup are the same as the BBR back up preparation steps.

Before using BBR you must perform the following steps:

- Verify your BBR Version
- Download the BBR SSH Credentials
- Download the BOSH Director Credentials
- Download the UAA Client Credentials
- Retrieve the BOSH Director Address
- Download the Root CA Certificate
- Download the BOSH Command Line Credentials

- Retrieve your Cluster Deployment Name

Verify Your BBR Version

Before running BBR, you must verify that the installed version of BBR is compatible with your deployment's current PKS release.

1. For your current PKS release's minimum version information, see the PKS Release Notes.
2. To verify the currently installed BBR version, run the following command:

   ```
   bbr version
   ```

   If you do not have BBR installed, or your installed version does not meet the minimum version requirement, see Installing BOSH Backup and Restore.

Collect Credentials and Account Information

Before you can perform a backup you will need to collect accounts and credentials to authenticate into your jumpbox, BOSH Director, and Ops Manager.

Download the BBR SSH Credentials

There are two ways to retrieve BOSH Director credentials:

- Ops Manager Installation Dashboard
- Ops Manager API

Ops Manager Installation Dashboard

To retrieve your **Bbr Ssh Credentials** using the Ops Manager Installation Dashboard, perform the following steps:

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the BOSH Director tile.
3. Click the **Credentials** tab.
4. Locate **Bbr Ssh Credentials**.
5. Click **Link to Credentials** next to it.
6. Copy the `private_key_pem` field value.

Ops Manager API

To retrieve your **Bbr Ssh Credentials** using the Ops Manager API, perform the following steps:

1. Obtain your UAA access token. For more information, see Access the API
2. Retrieve the **Bbr Ssh Credentials** by running the following command:

   ```
   curl "https://OPS-MAN-FQDN/api/v0/deployed/director/credentials/bbr_ssh_credentials" \
   -X GET \
   -H "Authorization: Bearer UAA-ACCESS-TOKEN"
   ```

   Where:

   - `OPS-MAN-FQDN` is the fully-qualified domain name (FQDN) for your Ops Manager deployment.
   - `UAA-ACCESS-TOKEN` is your UAA access token.

3. Copy the value of the `private_key_pem` field.

Save the BBR SSH Credentials to File

1. To reformat the copied `private_key_pem` value and save it to a file in the current directory, run the following command:

    ```
    printf -- "YOUR-PRIVATE-KEY" > PRIVATE-KEY-FILE
    ```

    Where:

    - `YOUR-PRIVATE-KEY` is the text of your private key.
    - `PRIVATE-KEY-FILE` is the path to the private key file you are creating.

    For example:

    ```
    $ printf --  "-----begin rsa private key----- fake key contents ----end rsa private key-----" > bbr_key.pem
    ```

Download the BOSH Director Credentials

There are two ways to retrieve BOSH Director credentials:

- Ops Manager Installation Dashboard
- Ops Manager API

Ops Manager Installation Dashboard

To retrieve your BOSH Director credentials using the Ops Manager Installation Dashboard, perform the following steps:

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the BOSH Director tile.
3. Click the **Credentials** tab.
4. Locate **Director Credentials**.
5. Click **Link to Credentials** next to it.
6. Copy the value of the `private_key_pem` field.

Ops Manager API

To retrieve your BOSH Director credentials using the Ops Manager API, perform the following steps:

1. Obtain your UAA access token. For more information, see Access the Ops Manager API
2. Retrieve the **Director Credentials** by running the following command:

    ```
    curl "https://OPS-MAN-FQDN/api/v0/deployed/director/credentials/bbr_ssh_credentials" \
    -X GET \
    -H "Authorization: Bearer UAA-ACCESS-TOKEN"
    ```

    Where: `OPS-MAN-FQDN` is the fully-qualified domain name (FQDN) for your Ops Manager deployment. `UAA-ACCESS-TOKEN` is your UAA access token.

3. Copy the value of the `private_key_pem` field.

Save the BOSH Director Credentials to a File

1. To reformat the copied `private_key_pem` value and save it a file in the current directory, run the following command:

```
printf -- "YOUR-PRIVATE-KEY" > PRIVATE-KEY-FILE
```

Where:

- `YOUR-PRIVATE-KEY` is the text of your private key.
- `PRIVATE-KEY-FILE` is the path to the private key file you are creating.

For example:

```
$ printf --  "-----begin rsa private key----- fake key contents ----end rsa private key-----" > bbr_key.pem
```

Download the UAA Client Credentials

To obtain BOSH credentials for your BBR operations, perform the following steps:

1. From the Ops Manager Installation Dashboard, click the **Pivotal Container Service** tile.
2. Select the **Credentials** tab.
3. Navigate to **Credentials > UAA Client Credentials**.
4. Record the value for `uaa_client_secret`.
5. Record the value for `uaa_client_name`.

Note: You must use BOSH credentials that limit the scope of BBR activity to your cluster deployments.

Retrieve the BOSH Director Address

You access the BOSH Director using an IP address.

To obtain your BOSH Director's IP address:

1. Open the Ops Manager Installation Dashboard.
2. Select **BOSH Director > Status**.
3. Select the listed Director IP Address.

Log In to BOSH Director

1. If you are not using the Ops Manager VM as your jumpbox, install the latest BOSH CLI on your jumpbox.
2. To log in to BOSH Director, using the IP address that you recorded above, run the following command line:

   ```
   bosh -e BOSH-DIRECTOR-IP \
   --ca-cert PATH-TO-BOSH-SERVER-CERTIFICATE log-in
   ```

   Where:

   - `BOSH-DIRECTOR-IP` is the BOSH Director IP address recorded above.
   - `PATH-TO-BOSH-SERVER-CERTIFICATE` is the path to the root Certificate Authority (CA) certificate as outlined in Download the Root CA Certificate.

3. To specify **Email**, specify `director`.
4. To specify **Password**, enter the **Director Credentials** that you obtained in Download the BOSH Director Credentials.
   For example:

```
$ bosh -e 10.0.0.3 \
--ca-cert /var/tempest/workspaces/default/root_ca_certificate log-in
Email (): director
Password (): ******************
Successfully authenticated with UAA
Succeeded
```

Download the Root CA Certificate

To download the root CA certificate for your PKS deployment, perform the following steps:

1. Open the Ops Manager Installation Dashboard
2. In the top right corner, click your username.
3. Navigate to **Settings** > **Advanced**.
4. Click **Download Root CA Cert**.

Download the BOSH Command Line Credentials

1. Open the Ops Manager Installation Dashboard.
2. Click the **BOSH Director** tile.
3. In the BOSH Director tile, click the **Credentials** tab.
4. Navigate to **Bosh Commandline Credentials**.
5. Click **Link to Credential**.
6. Copy the credential value.

Retrieve Your Cluster Deployment Name

To locate and record your cluster's BOSH deployment name, follow the steps below.

1. On the command line, run the following command to log in:

   ```
   pks login -a PKS-API -u USERNAME -k
   ```

   Where:

   - `PKS-API` is the domain name for the PKS API that you entered in **Ops Manager** > **Pivotal Container Service** > **PKS API** > **API Hostname (FQDN)**. For example, `api.pks.example.com`.
   - `USERNAME` is your user name.

   See Logging in to PKS for more information about the `pks login` command.

2. To find the cluster ID associated with the cluster you want to back up, run the following command:

   ```
   pks cluster CLUSTER-NAME
   ```

   Where `CLUSTER-NAME` is the name of your cluster.

3. From the output of this command, record the **UUID** value.
4. Open the Ops Manager Installation Dashboard
5. Click the **BOSH Director** tile.
6. Select the **Credentials** tab.

7. Navigate to **Bosh Commandline Credentials** and click **Link to Credential**.
8. Copy the credential value.
9. SSH into your jumpbox. For more information about the jumpbox, see Installing BOSH Backup and Restore.
10. To retrieve your cluster BOSH deployment name, run the following command:

    ```
    BOSH-CLI-CREDENTIALS deployments | grep UUID
    ```

    Where:

    - `BOSH-CLI-CREDENTIALS` is the full value that you copied from the BOSH Director tile in Download the BOSH Command Line Credentials.
    - `UUID` is the cluster UUID that you recorded in the previous step.

Transfer Artifacts to Your Jumpbox

To restore BOSH director, PKS control plane or cluster you must transfer your BBR backup artifacts from your safe storage location to your jumpbox.

1. To copy an artifact onto a jumpbox, run the following SCP command:

    ```
    scp -r LOCAL-PATH-TO-BACKUP-ARTIFACT JUMPBOX-USER@JUMPBOX-ADDRESS:
    ```

    Where:

    - `LOCAL-PATH-TO-BACKUP-ARTIFACT` is the path to your bbr backup artifact.
    - `JUMPBOX-USER` is the ssh username of the jumpbox.
    - `JUMPBOX-ADDRESS` is the IP address, or hostname, of the jumpbox.

2. (Optional) Decrypt your backup artifact if the artifact is encrypted.

Restore the BOSH Director

In the event of losing your BOSH Director or Ops Manager environment, you must first recreate the BOSH Director VM before restoring the BOSH Director.

You can restore your BOSH Director configuration by using PKS Ops Manager to restore the installation settings artifacts saved when following the Export Installation Settings backup procedure steps.

To redeploy and restore your Ops Manager and BOSH Director follow the procedures below.

Deploy Ops Manager

In the event of a disaster, you may lose your IaaS resources. You must recreate your IaaS resources before restoring using your BBR artifacts.

1. To recreate your IaaS resources, such as networks and load balancers, prepare your environment for PKS by following the installation instructions specific to your IaaS in Installing PKS.
2. After recreating IaaS resources, you must add those resources to Ops Manager by performing the procedures in the (Optional) Configure Ops Manager for New Resources section.

Import Installation Settings

⚠ **WARNING:** After importing installation settings, do not click **Apply Changes** in Ops Manager until instructed to in Redeploy the PKS Control Plane.

You can import installation settings in two ways:

- Use the Ops Manager UI:

1. Access your new Ops Manager by navigating to `YOUR-OPS-MAN-FQDN` in a browser.
2. On the **Welcome to Ops Manager** page, click **Import Existing Installation**.
3. In the import panel, perform the following tasks:
    - Enter the **Decryption Passphrase** in use when you exported the installation settings from Ops Manager.
    - Click **Choose File** and browse to the installation zip file that you exported in Back Up Installation Settings.
4. Click **Import**.

    Note: Some browsers do not provide the import process' progress status, and may appear to hang. The import process takes at least 10 minutes, and requires additional time for each restored Ops Manager tile.

5. **Successfully imported installation** is displayed upon successful completion of importing all installation settings.

- Use the Ops Manager API:

    1. To use the Ops Manager API to import installation settings, run the following command:

        ```
        curl "https://OPS-MAN-FQDN/api/v1/installation_asset_collection \
        -X POST \
        -H "Authorization: Bearer UAA-ACCESS-TOKEN" \
        -F 'installation[file]=@installation.zip' \
        -F 'passphrase=DECRYPTION-PASSPHRASE'
        ```

        Where:

        - `OPS-MAN-FQDN` is the fully-qualified domain name (FQDN) for your Ops Manager deployment.
        - `UAA-ACCESS-TOKEN` is the UAA access token. For more information about how to retrieve this token, see Using the Ops Manager API.
        - `DECRYPTION-PASSPHRASE` is the decryption passphrase in use when you exported the installation settings from Ops Manager.

(Optional) Configure Ops Manager for New Resources

If you recreated IaaS resources such as networks and load balancers by following the steps in the Deploy Ops Manager section above, perform the following steps to update Ops Manager with your new resources:

1. Enable Ops Manager advanced mode. For more information, see How to Enable Advanced Mode in the Ops Manager in the Pivotal Knowledge Base.

    Note: Ops Manager advanced mode allows you to make changes that are normally disabled. You may see warning messages when you save changes.

2. Navigate to the Ops Manager Installation Dashboard and click the BOSH Director tile.
3. If you are using Google Cloud Platform (GCP), click **Google Config** and update:
    a. **Project ID** to reflect the GCP project ID.
    b. **Default Deployment Tag** to reflect the environment name.
    c. **AuthJSON** to reflect the service account.
4. Click **Create Networks** and update the network names to reflect the network names for the new environment.
5. If your BOSH Director had an external hostname, you must change it in **Director Config > Director Hostname** to ensure it does not conflict with the hostname of the backed up Director.
6. Ensure that there are no outstanding warning messages in the BOSH Director tile, then disable Ops Manager advanced mode. For more information, see How to Enable Advanced Mode in the Ops Manager in the Pivotal Knowledge Base.

Note: A change in VM size or underlying hardware should not affect the ability for BBR restore data, as long as adequate storage space to restore the data exists.

Remove BOSH State File

1. SSH into your Ops Manager VM. For more information, see the Log in to the Ops Manager VM with SSH section of the *Advanced Troubleshooting with the BOSH CLI* topic.

2. To delete the `/var/tempest/workspaces/default/deployments/bosh-state.json` file, run the following on the Ops Manager VM:

```
sudo rm /var/tempest/workspaces/default/deployments/bosh-state.json
```

3. In a browser, navigate to your Ops Manager's fully-qualified domain name.

4. Log in to Ops Manager.

Deploy the BOSH Director

You can deploy the BOSH Director by itself in two ways:

- Use the Ops Manager UI:

    1. Open the Ops Manager Installation Dashboard.
    2. Click **Review Pending Changes**.
    3. On the Review Pending Changes page, click the **BOSH Director** checkbox.
    4. Click **Apply Changes**.

- Use the Ops Manager API:

    1. Use the Ops Manager API to deploy the BOSH Director.

Restore the BOSH Director

Restore the BOSH Director by running BBR commands on you jumpbox.

> **Note**: The BBR restore command can take a long time to complete. The example command in this section uses `nohup` and the restore process is run within your SSH session. If you instead run the BBR command in a `screen` or `tmux` session the task will run separately from your SSH session and will continue to run, even if your SSH connection to the jumpbox fails.

Perform the following steps to restore the BOSH Director:

1. Ensure the PKS BOSH Director backup artifact is in the folder from which you run BBR.

2. Run the BBR restore command to restore the PKS BOSH Director:

```
nohup bbr director  --host BOSH-DIRECTOR-IP \
--username bbr  --private-key-path PRIVATE-KEY-FILE \
restore \
--artifact-path PATH-TO-DIRECTOR-BACKUP
```

   Where:

   - `BOSH-DIRECTOR-IP` is the address of the BOSH Director. If the BOSH Director is public, BOSH-DIRECTOR-IP is a URL, such as `https://my-bosh.xxx.cf-app.com`. Otherwise, this is the internal IP `BOSH-DIRECTOR-IP` which you can retrieve as shown in Retrieve the BOSH Director Address.
   - `PRIVATE-KEY-FILE` is the path to the private key file that you can create from `Bbr Ssh Credentials` as shown in Download the BBR SSH Credentials.
   - `PATH-TO-DEPLOYMENT-BACKUP` is the path to the PKS BOSH Director backup that you want to restore.

   For example:

```
$ nohup bbr director  --host 10.0.0.5 \
  --username bbr  --private-key-path private.pem \
  restore \
  --artifact-path /home/10.0.0.5-abcd1234abcd1234
```

   If the command fails, follow the steps in Recover from a Failing Command.

   > **Note**: You can use the optional `--debug` flag to enable debug logs. See BBR Logging for more information.

Remove All Stale Deployment Cloud IDs

After BOSH Director has been restored, you must reconcile BOSH Director's internal state with the state of the IaaS.

1. To determine the existing deployments in your environment, run the following command:

   ```
   BOSH-CLI-CREDENTIALS bbr deployments
   ```

   Where:

   - `BOSH-CLI-CREDENTIALS` is the full `Bosh Commandline Credentials` value that you copied from the BOSH Director tile in Download the BOSH Commandline Credentials.

2. To reconcile the BOSH Director's internal state with the state of a single deployment, run the following command:

   ```
   BOSH-CLI-CREDENTIALS bosh -d DEPLOYMENT-NAME -n cck \
   --resolution delete_disk_reference \
   --resolution delete_vm_reference
   ```

   Where:

   - `BOSH-CLI-CREDENTIALS` is the full `Bosh Commandline Credentials` value that you copied from the BOSH Director tile in Download the BOSH Commandline Credentials.
   - `DEPLOYMENT-NAME` is a deployment name retrieved in the previous step.

3. Repeat the last command for each deployment in the IaaS.

Restore the PKS Control Plane

You must redeploy the PKS tile before restoring the PKS control plane. By redeploying the PKS tile you create the VMs that constitute the control plane deployment.

To redeploy the PKS tile, you must determine the stemcell needed by the tile, upload that stemcell, and restore the backup on top of the deployment.

Determine the Required Stemcells

Perform either the following procedures to determine which stemcell is used by PKS:

- Review the Stemcell Librbary:

  1. Open Ops Manager.
  2. Click **Stemcell Library**.
  3. Record the PKS stemcell release number from the **Staged** column.

- Review a Stemcell List Using BOSH CLI:

  1. To retrieve the stemcell release using the BOSH CLI, run the following command:

     ```
     BOSH-CLI-CREDENTIALS bosh deployments
     ```

     Where:

     - `BOSH-CLI-CREDENTIALS` is the full `Bosh Commandline Credentials` value that you copied from the BOSH Director tile in Download the BOSH Commandline Credentials.

     For example:

     ```
     $ bosh deployments
     Using environment '10.0.0.5' as user 'director' (bosh.*.read, openid, bosh.*.admin, bosh.read, bosh.admin)


     Name                                Release(s)                  Stemcell(s)                          Team(s)
     pivotal-container-service-453f2faa3bd2e16f52b7  backup-and-restore-sdk/1.8.0  bosh-google-kvm-ubuntu-xenial-go_agent/170.15  -
     ...
     ```

For more information about stemcells in Ops Manager, see Importing and Managing Stemcells.

Upload Stemcells

1. Download the stemcell from Pivotal Network.
2. Run the following command to upload the stemcell used by PKS:

```
BOSH-CLI-CREDENTIALS bosh -d DEPLOYMENT-NAME \
--ca-cert PATH-TO-BOSH-SERVER-CERTIFICATE \
upload-stemcell \
--fix PATH-TO-STEMCELL
```

   Where:

   - `BOSH-CLI-CREDENTIALS` is the full `Bosh Commandline Credentials` value that you copied from the BOSH Director tile in Download the BOSH Commandline Credentials.
   - `PATH-TO-BOSH-SERVER-CERTIFICATE` is the path to the root CA certificate that you downloaded in Download the Root CA Certificate.
   - `PATH-TO-STEMCELL` is the path to your tile's stemcell.

3. To ensure the stemcells for all of your other installed tiles have been uploaded, repeat the last step, running the `bosh upload-stemcell --fix PATH-TO-STEMCELL` command, for each required stemcell that is different from the already uploaded PKS stemcell.

Redeploy the PKS Control Plane

1. From the Ops Manager Installation Dashboard, navigate to **Pivotal Container Service** > **Resource Config**.
2. Ensure that all errands needed by your system are set to run.
3. Return to the Ops Manager Installation Dashboard.
4. Click **Review Pending Changes**.
5. Review your changes. For more information, see Reviewing Pending Product Changes.
6. Click **Apply Changes** to redeploy the control plane.

Restore the PKS Control Plane

Restore the PKS from your jumpbox.

> **Note**: The BBR restore command can take a long time to complete. The example command in this section uses `nohup` and the restore process is run within your SSH session. If you instead run the BBR command in a `screen` or `tmux` session the task will run separately from your SSH session and will continue to run, even if your SSH connection to the jumpbox fails.

To restore the PKS control plane perform the following steps:

1. Ensure the PKS deployment backup artifact is in the folder from which you run BBR.
2. Run the BBR restore command to restore the PKS control plane:

```
BOSH_CLIENT_SECRET=BOSH-CLIENT-SECRET \
nohup bbr deployment --target BOSH-TARGET \
--username BOSH-CLIENT --deployment DEPLOYMENT-NAME \
--ca-cert PATH-TO-BOSH-SERVER-CERT \
restore \
--artifact-path PATH-TO-DEPLOYMENT-BACKUP
```

   Where:

   - `BOSH-CLIENT-SECRET` is the value for `BOSH\_CLIENT\_SECRET` retrieved in Download the BOSH Commandline Credentials.
   - `BOSH-TARGET` is the value for `BOSH\_ENVIRONMENT` retrieved in Download the BOSH Commandline Credentials. You must be able to reach the target address from the workstation where you run `bbr` commands.
   - `BOSH-CLIENT` is the value for `BOSH_CLIENT` retrieved in Download the BOSH Commandline Credentials.
   - `DEPLOYMENT-NAME` is the deployment name retrieved in Locate the Enterprise PKS Deployment Name.
   - `PATH-TO-BOSH-CA-CERT` is the path to the root CA certificate that you downloaded in Download the Root CA Certificate.
   - `PATH-TO-DEPLOYMENT-BACKUP` is the path to the PKS control plane backup that you want to restore.

For example:

```
$ BOSH_CLIENT_SECRET=p455w0rd \
nohup bbr deployment  --target bosh.example.com \
--username admin  --deployment pivotal-container-0 \
--ca-cert bosh.ca.crt \
restore \
--artifact-path /home/pivotal-container-service_abcd1234abcd1234abcd-abcd1234abcd1234
```

If the command fails, follow the steps in Recover from a Failing Command.

Note: You can use the optional `–debug` flag to enable debug logs. See the BBR Logging topic for more information.

Restore PKS Clusters

After successfully restoring the PKS Control Plane, perform the following steps to restore the PKS clusters provisioned by the control plane.

Redeploy PKS Clusters

Before restoring, you must redeploy your PKS Clusters by performing the following steps:

1. Navigate to an Pivotal Container Service tile in the **Installation Dashboard**.
2. Select the **Errands** tab.
3. Ensure the **Upgrade all clusters** errand is **On**.
4. Return to the **Installation Dashboard**.
5. Click **Review Pending Changes**, review your changes, and then click **Apply Changes**. For more information, see Reviewing Pending Product Changes. This will include running the **Upgrade all clusters** errand, which will redeploy the cluster instances.

Restore Clusters

Perform the steps below to restore a cluster.

Note: The BBR restore command can take a long time to complete. The example command in this section uses `nohup` and the restore process is run within your SSH session. If you instead run the BBR command in a `screen` or `tmux` session the task will run separately from your SSH session and will continue to run, even if your SSH connection to the jumpbox fails.

Warning: When you restore the cluster, etcd is stopped in the API server. During this process, only currently-deployed clusters function, and you cannot create new workloads.

Warning: BBR only backs up and restores the cluster etcd data. This includes the cluster-deployed workloads. Persistent volumes and other IaaS resources, such as load balancers of workloads, are not backed up and restored. Backup and restore for clusters deployed on vSphere with NSX-T is not yet validated.

1. Move the cluster backup artifact to a folder from which you will run the BBR restore process.
2. SSH into your jumpbox. For more information about the jumpbox, see Configure Your Jumpbox in *Installing BOSH Backup and Restore*.
3. Run the following command:

```
BOSH_CLIENT_SECRET=BOSH-CLIENT-SECRET \
nohup bbr deployment  --target BOSH-TARGET \
--username BOSH-CLIENT  --deployment DEPLOYMENT-NAME \
--ca-cert PATH-TO-BOSH-SERVER-CERT \
restore \
--artifact-path PATH-TO-DEPLOYMENT-BACKUP
```

Where:

- `BOSH-CLIENT-SECRET` is the `BOSH_CLIENT_SECRET` property. This value is in the BOSH Director tile under **Credentials > Bosh Commandline Credentials**.
- `BOSH-TARGET` is the `BOSH_ENVIRONMENT` property. This value is in the BOSH Director tile under **Credentials > Bosh Commandline Credentials**. You must be able to reach the target address from the workstation where you run `bbr` commands.
- `BOSH-CLIENT` is the `BOSH_CLIENT` property. This value is in the BOSH Director tile under **Credentials > Bosh Commandline Credentials**.
- `DEPLOYMENT-NAME` is the cluster BOSH deployment name that you recorded in the Retrieve your Cluster Deployment Name section above.
- `PATH-TO-BOSH-CA-CERT` is the path to the root CA certificate that you downloaded in the Download the Root CA Certificate setcion above.
- `PATH-TO-DEPLOYMENT-BACKUP` is the path to to your deployment backup. Make sure you have transfer your artifact into you jumpbox as described in Transfer Artifacts to Jumpbox above.

For example:

```
$ BOSH_CLIENT_SECRET=p455w0rd \
nohup bbr deployment \
--target bosh.example.com \
--username admin \
--deployment service-instance_3839394 \
--ca-cert bosh.ca.cert \
restore \
--artifact-path deployment-backup
```

4. If the restore command fails, do one or more of the following:
   - Run the command again, adding the `--debug` flag to enable debug logs. For more information, see BBR Logging.
   - Follow the steps in Recover from a Failing Command below.

Recover from a Failing Command

1. Ensure that you set all the parameters in the command.
2. Ensure that the BOSH Director credentials are valid.
3. Ensure that the specified BOSH deployment or Director exists.
4. Ensure that the jumpbox can reach the BOSH Director.
5. Ensure the source backup artifact is compatible with the target BOSH deployment or Director.
6. If you see the error message `Directory /var/vcap/store/bbr-backup already exists on instance`, run the relevant commands from the Clean up After Failed Restore section of this topic.
7. See the BBR Logging topic.

Cancel a Restore

If you must cancel a restore, perform the following steps:

1. Terminate the BBR process by pressing Ctrl-C and typing `yes` to confirm.
2. Perform the procedures in the Clean up After Failed Restore section to enable future restores. Stopping a restore can leave the system in an unusable state and prevent future restores.

Clean Up After a Failed Restore

If a BBR restore process fails, BBR may not have run the post-restore scripts, potentially leaving the instance in a locked state. Additionally, the BBR restore folder may remain on the target instance and subsequent restore attempts may also fail.

- To resolve issues following a failed BOSH Director restore, run the following BBR command:

```
nohup bbr director \
--host BOSH-DIRECTOR-IP \
--username bbr \
--private-key-path PRIVATE-KEY-FILE \
restore-cleanup
```

Where:

- `BOSH-DIRECTOR-IP` is the address of the BOSH Director. If the BOSH Director is public, BOSH-DIRECTOR-IP is a URL, such as `https://my-bosh.xxx.cf-app.com`. Otherwise, this is the internal IP `BOSH-DIRECTOR-IP` which you can retrieve as show in Retrieve the BOSH Director Address above.
- `PRIVATE-KEY-FILE` is the path to the private key file that you can create from `Bbr Ssh Credentials` as shown in Download the BBR SSH Credentials above.
  For example:

  ```
  $ nohup bbr director \
  --target 10.0.0.5 \
  --username bbr \
  --private-key-path private.pem \
  restore-cleanup
  ```

- To resolve issues following a failed control plane restore, run the following BBR command:

  ```
  BOSH_CLIENT_SECRET=BOSH-CLIENT-SECRET \
  bbr deployment \
  --target BOSH-TARGET \
  --username BOSH-CLIENT \
  --deployment DEPLOYMENT-NAME \
  --ca-cert PATH-TO-BOSH-CA-CERT \
  restore-cleanup
  ```

  Where:

  - `BOSH-CLIENT-SECRET` is the value for `BOSH\_CLIENT\_SECRET` retrieved in Download the BOSH Commandline Credentials above.
  - `BOSH-TARGET` is the value for `BOSH\_ENVIRONMENT` retrieved in Download the BOSH Commandline Credentials above.
    You must be able to reach the target address from the workstation where you run `bbr` commands.
  - `BOSH-CLIENT` is the value for `BOSH_CLIENT` retrieved in Download the BOSH Commandline Credentials above.
  - `DEPLOYMENT-NAME` is the name retrieved in Retrieve Your Cluster Deployment Name above.
  - `PATH-TO-BOSH-CA-CERT` is the path to the root CA certificate that you downloaded in Download the Root CA Certificate above.
    For example:

    ```
    $ BOSH_CLIENT_SECRET=p455w0rd \
    bbr deployment \
    --target bosh.example.com \
    --username admin \
    --deployment pivotal-container-service-453f2f \
    --ca-cert bosh.ca.crt \
    restore-cleanup
    ```

- To resolve issues following a failed cluster restore, run the following BBR command:

  ```
  BOSH_CLIENT_SECRET=BOSH-CLIENT-SECRET \
  bbr deployment \
  --target BOSH-TARGET \
  --username BOSH-CLIENT \
  --deployment DEPLOYMENT-NAME \
  --ca-cert PATH-TO-BOSH-CA-CERT \
  restore-cleanup
  ```

  Where:

  - `BOSH-CLIENT-SECRET` is the value for `BOSH\_CLIENT\_SECRET` retrieved in Download the BOSH Commandline Credentials.
  - `BOSH-TARGET` is the value for `BOSH\_ENVIRONMENT` retrieved in Download the BOSH Commandline Credentials.
    You must be able to reach the target address from the workstation where you run `bbr` commands.
  - `BOSH-CLIENT` is the value for `BOSH_CLIENT` retrieved in Download the BOSH Commandline Credentials.
  - `DEPLOYMENT-NAME` is the name retrieved in Retrieve Your Cluster Deployment Name above.
  - `PATH-TO-BOSH-CA-CERT` is the path to the root CA certificate that you downloaded in Download the Root CA Certificate.
    For example:

```
$ BOSH_CLIENT_SECRET=p455w0rd \
bbr deployment \
--target bosh.example.com \
--username admin \
--deployment pivotal-container-service-453f2f \
--ca-cert bosh.ca.crt \
restore-cleanup
```

Please send any feedback you have to pks-feedback@pivotal.io.

BBR Logging

This topic provides information about BBR logging. Use this information when troubleshooting a failed backup or restore using BBR.

Understand Logging

By default, BBR displays the following:

- The backup and restore scripts that it finds
- When it starts or finishes a stage, such as `pre-backup scripts` or `backup scripts`
- When the process is complete
- When any error occurs

BBR writes any errors associated with stack traces to a file in of the form `bbr-TIMESTAMP.err.log` in the current directory.

If more logging is needed, use the optional `--debug` flag to print the following information:

- Logs about the API requests made to the BOSH server
- All commands executed on remote instances
- All commands executed on local environment
- Standard in and standard out streams for the backup and restore scripts when they are executed

---

Please send any feedback you have to pks-feedback@pivotal.io.

PKS Security

Page last updated:

This section includes security topics for Pivotal Container Service (PKS).

See the following topic:

- PKS Security Disclosure and Release Process

Please send any feedback you have to pks-feedback@pivotal.io.

PKS Security Disclosure and Release Process

Page last updated:

This topic describes the processes for disclosing security issues and releasing related fixes for Pivotal Container Service (PKS), Kubernetes, Cloud Foundry Container Runtime (CFCR), VMware NSX, and VMware Harbor.

Security Issues in PKS

Pivotal and VMware provide security coverage for PKS. Please report any vulnerabilities directly to Pivotal Application Security Team or the VMware Security Response Center.

Security fixes are provided in accordance with the PCF Security Release Policy and the Pivotal Support Lifecycle Policy.

Where applicable, security issues may be coordinated with the responsible disclosure process for the open source security teams in Kubernetes and Cloud Foundry projects.

Security Issues in Kubernetes

Pivotal and VMware follow the Kubernetes responsible disclosure process to work within the Kubernetes project to report and address suspected security issues with Kubernetes.

This process is discussed in Kubernetes Security and Disclosure Information.

When the Kubernetes project releases security fixes, PKS releases fixes according to the PCF Security Release Policy and the Pivotal Support Lifecycle Policy.

Security Issues in CFCR

Pivotal and VMware follow the Cloud Foundry responsible disclosure process to work within the Cloud Foundry Foundation to report and address suspected security issues with CFCR.

This process is discussed in Cloud Foundry Security.

When the Cloud Foundry Foundation releases security fixes, PKS releases fixes according to the PCF Security Release Policy and the Pivotal Support Lifecycle Policy.

Security Issues in VMware NSX

Security issues in VMware NSX are coordinated with the VMware Security Response Center.

Security Issues in VMware Harbor

Security issues in VMware Harbor are coordinated with the VMware Security Response Center.

Please send any feedback you have to pks-feedback@pivotal.io.

Diagnosing and Troubleshooting PKS

This topic is intended to provide assistance when diagnosing and troubleshooting issues installing or using Pivotal Container Service (PKS).

See the following sections:

Please send any feedback you have to pks-feedback@pivotal.io.

Diagnostic Tools

Verify PKS CLI Version

The Pivotal Container Service (PKS) CLI interacts with the your PKS deployment through the PKS API endpoint. You create, manage, and delete Kubernetes clusters on your PKS deployment by entering commands in the PKS CLI. The PKS CLI is under active development and commands may change between versions.

To determine the version of PKS CLI installed locally, run the following command:

```
pks --version
```

For example:

```
$ pks --version
PKS CLI version: 1.0.0-build.3
```

SSH into the PKS VM

To SSH into the PKS VM using BOSH, follow the steps below:

1. Gather credential and IP address information for your BOSH Director, SSH into the Ops Manager VM, and use BOSH CLI to log in to the BOSH Director from the Ops Manager VM. For more information, see Advanced Troubleshooting with the BOSH CLI.

2. To identify your PKS deployment's name, run the following command:

   ```
   bosh -e ENVIRONMENT deployments
   ```

   Where `ENVIRONMENT` is the BOSH environment alias you set in Set a BOSH Environment Alias.

   For example:

   ```
   $ bosh -e pks deployments
   ```

   Your PKS deployment name begins with `pivotal-container-service` and includes a BOSH-generated hash.

3. To identify your PKS VM's name, run the following command:

   ```
   bosh -e ENVIRONMENT -d DEPLOYMENT vms
   ```

   Where:

   - `ENVIRONMENT` is the BOSH environment alias.
   - `DEPLOYMENT` is your PKS deployment name.

   For example:

   ```
   $ bosh -e pks -d pivotal-container-service/a1b2c333d444e5f66a77 vms
   ```

   Your PKS VM name begins with `pivotal-container-service` and includes a BOSH-generated hash.

   Note: The PKS VM hash value is different from the hash in your PKS deployment name.

4. To SSH into the PKS VM, run the following command:

   ```
   bosh -e ENVIRONMENT -d DEPLOYMENT ssh PKS-VM
   ```

   Where:

   - `ENVIRONMENT` is the BOSH environment alias.

- `DEPLOYMENT` is your PKS deployment name.
- `PKS-VM` is your PKS VM name.

For example:

```
$ bosh -e pks \
-d pivotal-container-service/a1b2c333d444e5f66a77 \
ssh pivotal-container-service/000a1111-222b-3333-4cc5-de66f7a8899b
```

SSH into the Kubernetes Cluster Master Node VM

To SSH into the master node VM for a PKS Kubernetes cluster using BOSH, follow the steps below:

1. Gather credential and IP address information for your BOSH Director, SSH into the Ops Manager VM, and use BOSH CLI to log in to the BOSH Director from the Ops Manager VM. For more information, see Advanced Troubleshooting with the BOSH CLI.

2. To identify your PKS deployment's name, run the following command:

   ```
   bosh -e ENVIRONMENT deployments
   ```

   Where `ENVIRONMENT` is the BOSH environment alias you set in the Set a BOSH Environment Alias section of *Managing PKS Deployments with BOSH*.

   For example:

   ```
   $ bosh -e pks deployments
   ```

   Your PKS deployment name begins with `pivotal-container-service` and includes a BOSH-generated hash.

3. To identify your PKS VM's name, run the following command:

   ```
   bosh -e ENVIRONMENT -d DEPLOYMENT vms
   ```

   Where:

   - `ENVIRONMENT` is the BOSH environment alias.
   - `DEPLOYMENT` is your PKS deployment name.

   For example:

   ```
   $ bosh -e pks -d service-instance_ae681cd1-7ff4-4661-b12c-49a5b543f16f vms
   ```

   Your PKS VM name begins with `pivotal-container-service` and includes a BOSH-generated hash.

   Note: The PKS VM hash value is different from the hash in your PKS deployment name.

4. To SSH into the master node of the cluster, run the following command:

   ```
   bosh -e ENVIRONMENT -d DEPLOYMENT ssh master/MASTER-NUMBER
   ```

   Where:

   - `ENVIRONMENT` is the BOSH environment alias.
   - `DEPLOYMENT` is your PKS deployment name.
   - `MASTER-NUMBER` is your master number.

   For example:

   ```
   $ bosh -e pks -d service-instance_ae681cd1-7ff4-4661-b12c-49a5b543f16f ssh master/0
   ```

View Log Files

Log files contain error messages and other information you can use to diagnose issues with your PKS deployment. SSH into the PKS VM then follow the steps below to access PKS log files.

1. To act as super user on the PKS VM, run the following command:

    ```
    sudo su
    ```

2. To navigate to the PKS VM's `/var/vcap/sys/log` log file directory, run the following command:

    ```
    cd /var/vcap/sys/log
    ```

3. Examine the following files:

    - On the PKS master VM, examine the `kube-apiserver` log file.
    - On a PKS worker VM, examine the `kubelet` log file.

Please send any feedback you have to pks-feedback@pivotal.io.

Verifying Deployment Health

Page last updated:

This topic describes how to verify the health of your Pivotal Container Service (PKS) deployment.

Verify Kubernetes Health

Verify the health of your Kubernetes environment by following the steps below:

1. From the Ops Manager VM, run the following command:

   ```
   bosh -e ENV-NAME login
   ```

   Where `ENV-NAME` is the alias you set for your BOSH Director. For more information, see Managing PKS Deployments with BOSH.

   For example:

   ```
   $ bosh -e pks login
   ```

2. To verify that all nodes are in a ready state, run the following command for all Kubernetes contexts:

   ```
   kubectl get nodes
   ```

3. To verify that all pods are running, run the following command for all Kubernetes contexts:

   ```
   kubectl get pods --all-namespaces
   ```

4. To get the name of the target Kubernetes deployment, run the following command:

   ```
   bosh deployments
   ```

   For example:

```
$ bosh deployments
Using environment '30.0.0.10' as client 'ops_manager'

Name                                                Release(s)                      Stemcell(s)                                  Team(s)
harbor-container-registry-b4023f6857207b237399      bosh-dns/1.10.0                 bosh-vsphere-esxi-ubuntu-xenial-go_agent/170.15  -
                                                    harbor-container-registry/1.7.3-build.2
pivotal-container-service-7e64d53fc570503b5690      backup-and-restore-sdk/1.8.0    bosh-vsphere-esxi-ubuntu-xenial-go_agent/170.15  -
                                                    bosh-dns/1.10.0
                                                    bpm/0.13.0
                                                    cf-mysql/36.14.0
                                                    cfcr-etcd/1.8.0
                                                    docker/35.0.0
                                                    harbor-container-registry/1.7.3-build.2
                                                    kubo/0.25.9
                                                    kubo-service-adapter/1.3.3-build.1
                                                    nsx-cf-cni/2.3.1.10693410
                                                    on-demand-service-broker/0.24.0
                                                    pks-api/1.3.3-build.1
                                                    pks-helpers/50.0.0
                                                    pks-nsx-t/1.19.0
                                                    pks-telemetry/2.0.0-build.113
                                                    pks-vrli/0.7.0
                                                    sink-resources-release/0.1.15
                                                    syslog/11.4.0
                                                    uaa/64.0
                                                    wavefront-proxy/0.9.0
service-instance_8de000ff-a87a-4930-81ba-106d42c2471e  bosh-dns/1.10.0              bosh-vsphere-esxi-ubuntu-xenial-go_agent/170.15  pivotal-container-service-7e64d53fc570503
                                                    bpm/0.13.0
                                                    cfcr-etcd/1.8.0
                                                    docker/35.0.0
                                                    harbor-container-registry/1.7.3-build.2
                                                    kubo/0.25.9
                                                    nsx-cf-cni/2.3.1.10693410
                                                    pks-helpers/50.0.0
                                                    pks-nsx-t/1.19.0
                                                    pks-telemetry/2.0.0-build.113
                                                    pks-vrli/0.7.0
                                                    sink-resources-release/0.1.15
                                                    syslog/11.4.0
                                                    wavefront-proxy/0.9.0

3 deployments
```

In the example above, `service-instance_8de000ff-a87a-4930-81ba-106d42c2471e` is the Kubernetes deployment name.

> **Note**: If you have deployed multiple Kubernetes clusters, determine the UUID using `pks clusters`, then match that UUID with the Kubernetes cluster deployment you are targeting.

5. To assess the status of the VMs comprising the target Kubernetes cluster, run the following command:

```
bosh -d K8-DEPLOYMENT vms
```

Where `K8-DEPLOYMENT` is the name of your Kubernetes cluster deployment. Kubernetes cluster deployment names begin with `service-instance_` and include a unique BOSH-generated hash.

This command returns the name of each VM comprising the Kubernetes cluster, including each master and worker node.

For example:

```
$ bosh -d service-instance_8de000ff-a87a-4930-81ba-106d42c2471e vms
Using environment '30.0.0.10' as client 'ops_manager'


Task 677. Done


Deployment 'service-instance_8de000ff-a87a-4930-81ba-106d42c2471e'


Instance                                      Process State  AZ      IPs       VM CID                                   VM Type      Active
master/b6d3c263-1682-4c79-a9ab-35939127dedb   running        AZ-K8S  40.0.2.2  vm-60dbcf68-5538-4c4e-8c00-61edc003bb54  medium.disk  true
worker/d450548a-2b0c-4494-8144-cf9b7ef9c825   running        AZ-K8S  40.0.2.4  vm-1bfdde6d-ce1d-4cdf-90d9-32bba260358f  medium.disk  true
worker/d7f882f0-33dd-43d3-ab5d-058bcc505088   running        AZ-K8S  40.0.2.3  vm-822cb573-411f-4c44-a32b-34e79520a7a6  medium.disk  true
worker/e5e25ffe-f448-4d19-990b-89546118c502   running        AZ-K8S  40.0.2.5  vm-c6748604-8440-4b27-9cf4-10a70a02da24  medium.disk  true


4 vms


Succeeded
```

6. To verify that all the processes in the Kubernetes cluster are in a running state, run the following command for each deployment:

   ```
   bosh -d K8-DEPLOYMENT instances --ps
   ```

   Where `K8-DEPLOYMENT` is the name of your Kubernetes cluster deployment. Kubernetes cluster deployment names begin with `service-instance_` and include a unique BOSH-generated hash.

   For example:

```
$ bosh -d service-instance_8de000ff-a87a-4930-81ba-106d42c2471e instances --ps
Using environment '30.0.0.10' as client 'ops_manager'


Task 678. Done


Deployment 'service-instance_8de000ff-a87a-4930-81ba-106d42c2471e'


Instance                                     Process                    Process State  AZ     IPs
apply-addons/ef0d09ae-d3ed-4832-8d3f-431d99730c26  -                          -              AZ-K8S  -
master/b6d3c263-1682-4c79-a9ab-35939127dedb        -                          running        AZ-K8S  40.0.2.2
~                                            blackbox                   running        -      -
~                                            bosh-dns                   running        -      -
~                                            bosh-dns-healthcheck       running        -      -
~                                            bosh-dns-resolvconf        running        -      -
~                                            etcd                       running        -      -
~                                            kube-apiserver             running        -      -
~                                            kube-controller-manager    running        -      -
~                                            kube-scheduler             running        -      -
~                                            ncp                        running        -      -
~                                            pks-helpers-bosh-dns-resolvconf  running        -      -
worker/d450548a-2b0c-4494-8144-cf9b7ef9c825        -                          running        AZ-K8S  40.0.2.4
~                                            blackbox                   running        -      -
~                                            bosh-dns                   running        -      -
~                                            bosh-dns-healthcheck       running        -      -
~                                            bosh-dns-resolvconf        running        -      -
~                                            docker                     running        -      -
~                                            kube-proxy                 running        -      -
~                                            kubelet                    running        -      -
~                                            nsx-kube-proxy             running        -      -
~                                            nsx-node-agent             running        -      -
~                                            ovs-vswitchd               running        -      -
~                                            ovsdb-server               running        -      -
~                                            pks-helpers-bosh-dns-resolvconf  running        -      -
worker/d7f882f0-33dd-43d3-ab5d-058bcc505088        -                          running        AZ-K8S  40.0.2.3
~                                            blackbox                   running        -      -
~                                            bosh-dns                   running        -      -
~                                            bosh-dns-healthcheck       running        -      -
~                                            bosh-dns-resolvconf        running        -      -
~                                            docker                     running        -      -
~                                            kube-proxy                 running        -      -
~                                            kubelet                    running        -      -
~                                            nsx-kube-proxy             running        -      -
~                                            nsx-node-agent             running        -      -
~                                            ovs-vswitchd               running        -      -
~                                            ovsdb-server               running        -      -
~                                            pks-helpers-bosh-dns-resolvconf  running        -      -
worker/e5e25ffe-f448-4d19-990b-89546118c502        -                          running        AZ-K8S  40.0.2.5
~                                            blackbox                   running        -      -
~                                            bosh-dns                   running        -      -
~                                            bosh-dns-healthcheck       running        -      -
~                                            bosh-dns-resolvconf        running        -      -
~                                            docker                     running        -      -
~                                            kube-proxy                 running        -      -
~                                            kubelet                    running        -      -
~                                            nsx-kube-proxy             running        -      -
~                                            nsx-node-agent             running        -      -
~                                            ovs-vswitchd               running        -      -
~                                            ovsdb-server               running        -      -
~                                            pks-helpers-bosh-dns-resolvconf  running        -      -


51 instances
```

Verify NCP Health (NSX-T Only)

NCP runs as a BOSH host process. Each Kubernetes master node VM has one running NCP process. If your cluster has multiple master nodes, one NCP process is active while the others are on standby. For more information, see Architectural Changes.

Verify NCP health by following the steps below:

1. Perform the steps in Verify Kubernetes Health to view the VMs and process instances for your target Kubernetes cluster.
2. To verify the `ncp` process is `running`, re-run the bosh `instances` command on the master node:

```
bosh -d K8-DEPLOYMENT instances --ps
```

Where `K8-DEPLOYMENT` is the name of your Kubernetes cluster deployment. Kubernetes cluster deployment names begin with `service-instance_` and include a unique BOSH-generated hash.

For example:

```
$ bosh -d service-instance_8de000ff-a87a-4930-81ba-106d42c2471e instances --ps
Using environment '30.0.0.10' as client 'ops_manager'

Task 678. Done

Deployment 'service-instance_8de000ff-a87a-4930-81ba-106d42c2471e'

Instance                                Process                      Process State  AZ      IPs
apply-addons/ef0d09ae-d3ed-4832-8d3f-431d99730c26  -                         -            AZ-K8S  -
master/b6d3c263-1682-4c79-a9ab-35939127dedb        -                         running      AZ-K8S  40.0.2.2
~                                       blackbox                     running      -       -
~                                       bosh-dns                     running      -       -
~                                       bosh-dns-healthcheck         running      -       -
~                                       bosh-dns-resolvconf          running      -       -
~                                       etcd                         running      -       -
~                                       kube-apiserver               running      -       -
~                                       kube-controller-manager      running      -       -
~                                       kube-scheduler               running      -       -
~                                       ncp                          running      -       -
~                                       pks-helpers-bosh-dns-resolvconf  running      -       -
```

3. To SSH into the Kubernetes master node VM, run the following command:

   ```
   bosh -e ENV-NAME -d PKS-DEPLOYMENT ssh VM-NAME/VM-ID
   ```

   Where:

   - `ENV-NAME` is the alias you set for your BOSH Director. For more information, see Managing PKS Deployments with BOSH.
   - `PKS-DEPLOYMENT` is the name of your PKS installation. PKS deployment names begin with `pivotal-container-service` and include a unique BOSH-generated hash.
   - `VM-NAME` is your Kubernetes master node VM name.
   - `VM-ID` is your Kubernetes master node VM ID. This is a unique BOSH-generated hash.

     For example:

     ```
     $ bosh -e pks -d pivotal-container-service-7e64d53fc570503b5690 ssh master/b6d3c263-1682-4c79-a9ab-35939127dedb
     ```

4. To verify the `ncp` process is `running`, run the following command from the master node VM:

   ```
   monit summary
   ```

5. To check if the NCP process is active or on standby, run the following command:

   ```
   /var/vcap/jobs/ncp/bin/nsxcli -c get ncp-master status
   ```

6. To restart the NCP process, run the following command:

   ```
   monit restart ncp
   ```

7. To verify that the NCP process restarts successfully, run the following command:

   ```
   monit summary
   ```

Service Interruptions

Page last updated:

This topic describes events in the lifecycle of a Kubernetes cluster deployed by Pivotal Container Service (PKS) that can cause temporary service interruptions.

Stemcell or Service Update

An operator updates the stemcell version or PKS version.

Impact

- **Workload**: If you run the recommended configuration, no workload downtime is expected since the VMs are upgraded one at a time. For more information, see Maintaining Workload Uptime.
- **Kubernetes control plane**: The Kubernetes master VM is recreated during the upgrade, so `kubectl` and the Kubernetes control plane experience a short downtime.

Required Actions

None. If the update deploys successfully, the Kubernetes control plane recovers automatically.

VM Process Failure on a Cluster Master

A process, such as the scheduler or the Kubernetes API server, crashes on the cluster master VM.

Impact

- **Workload**: If the scheduler crashes, workloads that are in the process of being rescheduled may experience up to 120 seconds of downtime.
- **Kubernetes control plane**: Depending on the process and what it was doing when it crashed, the Kubernetes control plane may experience 60-120 seconds of downtime. Until the process resumes, the following can occur:
  - Developers may be unable to deploy workloads
  - Metrics or logging may stop
  - Other features may be interrupted

Required Actions

None. BOSH brings the process back automatically using `monit`. If the process resumes cleanly and without manual intervention, the Kubernetes control plane recovers automatically.

VM Process Failure on a Cluster Worker

A process, such as Docker or `kube-proxy`, crashes on a cluster worker VM.

Impact

- **Workload**: If the cluster and workloads follow the recommended configuration for the number of workers, replica sets, and pod anti-affinity rules, workloads should not experience downtime. The Kubernetes scheduler reschedules the affected pods on other workers. For more information, see Maintaining Workload Uptime.

Required Actions

None. BOSH brings the process back automatically using `monit`. If the process resumes cleanly and without manual intervention, the worker recovers automatically, and the scheduler resumes scheduling new pods on this worker.

VM Process Failure on the Pivotal Container Service VM

A process, such as the PKS API server, crashes on the pivotal-container-service VM.

Impact

- **PKS control plane**: Depending on the process and what it was doing, the PKS control plane may experience 60-120 seconds of downtime. Until the process resumes, the following can occur:
  - The PKS API or UAA may be inaccessible
  - Use of the PKS CLI is interrupted
  - Metrics or logging may stop
  - Other features may be interrupted

Required Actions

None. BOSH brings the process back automatically using `monit`. If the process resumes cleanly, the PKS control plane recovers automatically and the PKS CLI resumes working.

VM Failure

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

Impact

- **If the BOSH Resurrector is enabled**, BOSH detects the failure, recreates the VM, and reattaches the same persistent disk and IP address. Downtime depends on which VM goes offline, how quickly the BOSH Resurrector notices, and how long it takes the IaaS to create a replacement VM. The BOSH Resurrector usually notices an offline VM within one to two minutes. For more information about the BOSH Resurrector, see the BOSH documentation.
- **If the BOSH Resurrector is not enabled**, some cloud providers, such as vSphere, have similar resurrection or high availability (HA) features. Depending on the VM, the impact can be similar to a key process on that VM going down as described in the previous sections, but the recovery time is longer while the replacement VM is created. See the sections for process failures on the cluster worker, cluster master, and PKS VM sections for more information.

Required Actions

When the VM comes back online, no further action is required for the developer to continue operations.

AZ Failure

An availability zone (AZ) goes offline entirely or loses connectivity to other AZs (net split).

Impact

The control plane and clusters are inaccessible. The extent of the downtime is unknown.

Required Actions

When the AZ comes back online, the control plane recovers in one of the following ways:

- **If BOSH is in a different AZ**, BOSH recreates the VMs with the last known persistent disks and IPs. If the persistent disks are gone, the disks can be restored from your last backup and reattached. Pivotal recommends manually checking the state of VMs and databases.
- **If BOSH is in the same AZ**, follow the directions for region failure.

Region Failure

An entire region fails, bringing all PKS components offline.

Impact

The entire PKS deployment and all services are unavailable. The extent of the downtime is unknown.

Required Actions

The PKS control plane can be restored using BOSH Backup and Restore (BBR). Each cluster may need to be restored manually from backups.

For more information, see Restoring the PKS Control Plane.

Please send any feedback you have to pks-feedback@pivotal.io.

Troubleshooting

Page last updated:

PKS API is Slow or Times Out

Symptom

When you run PKS CLI commands, the PKS API times out or is slow to respond.

Explanation

The PKS API control plane VM requires more resources.

Solution

1. Navigate to `https://YOUR-OPS-MANAGER-FQDN/` in a browser to log in to the Ops Manager Installation Dashboard.
2. Select the **Pivotal Container Service** tile.
3. Select the **Resource Config** page.
4. For the **Pivotal Container Service** job, select a **VM Type** with greater CPU and memory resources.
5. Click **Save**.
6. Click the **Installation Dashboard** link to return to the Installation Dashboard.
7. Click **Review Pending Changes**. Review the changes that you made. For more information, see Reviewing Pending Product Changes.
8. Click **Apply Changes**.

All Cluster Operations Fail

Symptom

All PKS CLI cluster operations fail including attempts to create or delete clusters with `pks create-cluster` and `pks delete-cluster`.

The output of `pks cluster CLUSTER-NAME` contains `Last Action State: error`, and the output of `bosh -e ENV-ALIAS -d SERVICE-INSTANCE vms` indicates that the `Process State` of at least one deployed node is `failing`.

Explanation

If any deployed master or worker nodes run out of disk space in `/var/vcap/store`, all cluster operations such as the creation or deletion of clusters will fail.

Diagnostics

To confirm that there is a disk space issue, check recent BOSH activity for any disk space error messages.

1. Log in to the BOSH Director and run `bosh tasks`. The output from `bosh tasks` provides details about the tasks that the BOSH Director has run. See Managing PKS Deployments with BOSH for more information about logging in to the BOSH Director.
2. In the BOSH command output, locate a task that attempted to perform a cluster operation, such as cluster creation or deletion.
3. To retrieve more information about the task, run the following command:

   ```
   bosh -e MY-ENVIRONMENT task TASK-NUMBER
   ```

   Where:

   - `MY-ENVIRONMENT` is the name of your BOSH environment.
   - `TASK-NUMBER` is the number of the task that attempted to create the cluster.

For example:

```
$ bosh -e pks task 23
```

4. In the output, look for the following text string:

   ```
   no space left on device
   ```

5. Check the health of your deployed Kubernetes clusters by following the procedure in Verifying Deployment Health.

6. In the output of `bosh -e ENV-ALIAS -d SERVICE-INSTANCE vms`, look for any nodes that display `failing` as their `Process State`. For example:

   ```
   Instance                                     Process State  AZ  IPs       VM CID                                   VM Type  Active
   master/3a3adc92-14ce-4cd4-a12c-6b5eb03e33d6  failing        az-1  10.0.11.10  vm-09027f0e-dac5-498e-474e-b47f2cda614d  small    true
   ```

7. Make a note of the plan assigned to the failing node.

Solution

1. In the PKS tile, locate the plan assigned to the failing node.

2. In the plan configuration, select a larger VM type for the plan's master or worker nodes or both.
   For more information about scaling existing clusters by changing the VM types, see Scale Vertically by Changing Cluster Node VM Sizes in the PKS Tile.

Cluster Creation Fails

Symptom

When creating a cluster, you run `pks cluster CLUSTER-NAME` to monitor the cluster creation status. In the command output, the value for **Last Action State** is `error`.

Explanation

There was an error creating the cluster.

Diagnostics

1. Log in to the BOSH Director and run `bosh tasks`. The output from `bosh tasks` provides details about the tasks that the BOSH Director has run. See Managing PKS Deployments with BOSH for more information about logging in to the BOSH Director.

2. In the BOSH command output, locate the task that attempted to create the cluster.

3. To retrieve more information about the task, run the following command:

   ```
   bosh -e MY-ENVIRONMENT task TASK-NUMBER
   ```

   Where:

   - `MY-ENVIRONMENT` is the name of your BOSH environment.
   - `TASK-NUMBER` is the number of the task that attempted to create the cluster.
     For example:

     ```
     $ bosh -e pks task 23
     ```

BOSH logs are used for error diagnostics but if the issue you see in the BOSH logs is related to using or managing Kubernetes, you should consult the Kubernetes Documentation for troubleshooting that issue.

For troubleshooting failed BOSH tasks, see the BOSH documentation.

Cluster Deletion Fails

Symptom

When attempting to delete a cluster using `pks delete-cluster CLUSTER-NAME`, the cluster is not deleted.

Explanation

There was an error deleting the cluster.

Solution

Log in to the BOSH Director and delete the BOSH deployment manually, then retry the `pks delete-cluster` operation.

1. Log in to the BOSH Director and obtain the deployment name for cluster you want to delete. For instructions, see Managing PKS Deployments with BOSH.
2. Run the following BOSH command:

   ```
   bosh -e MY-ENVIRONMENT delete-deployment -d DEPLOYMENT-NAME
   ```

   Where:

   - `MY-ENVIRONMENT` is the name of your BOSH environment.
   - `DEPLOYMENT-NAME` is the name of your BOSH deployment.

   > **Note**: If necessary, you can append the `--force` flag to delete the deployment.

3. Run the following PKS command:

   ```
   pks delete-cluster CLUSTER-NAME
   ```

   Where `CLUSTER-NAME` is the name of your PKS cluster.

Cannot Re-Create a Cluster that Failed to Deploy

Symptom

After cluster creation fails, you cannot re-run `pks create-cluster` to attempt creating the cluster again.

Explanation

PKS does not automatically clean up the failed BOSH deployment. Running `pks create-cluster` using the same cluster name creates a name clash error in BOSH.

Solution

Log in to the BOSH Director and delete the BOSH deployment manually, then retry the `pks delete-cluster` operation. After cluster deletion succeeds, re-create the cluster.

1. Log in to the BOSH Director and obtain the deployment name for cluster you want to delete. For instructions, see Managing PKS Deployments with BOSH.
2. Run the following BOSH command:

   ```
   bosh -e MY-ENVIRONMENT delete-deployment -d DEPLOYMENT-NAME
   ```

   Where:

   - `MY-ENVIRONMENT` is the name of your BOSH environment.
   - `DEPLOYMENT-NAME` is the name of your BOSH deployment.

   > **Note**: If necessary, you can append the `--force` flag to delete the deployment.

3. Run the following PKS command:

```
pks delete-cluster CLUSTER-NAME
```

Where `CLUSTER-NAME` is the name of your PKS cluster.

4. To re-create the cluster, run the following PKS command:

```
pks create-cluster CLUSTER-NAME
```

Where `CLUSTER-NAME` is the name of your PKS cluster.

Cannot Access Add-On Features or Functions

Symptom

You cannot access a feature or function provided by a Kubernetes add-on.

Examples include the following:

- You cannot access the Kubernetes Web UI (Dashboard) in a browser or using the kubectl command-line tool.
- Pods cannot resolve DNS names, and error messages report the service `kube-dns` is invalid. If `kube-dns` is not deployed, the cluster typically fails to start.
- Heapster does not start.

Explanation

The Kubernetes features and functions listed above are provided by the following PKS add-ons:

- **Kubernetes Dashboard** `kubernetes-dashboard`
- **DNS Resolution**: `kube-dns`
- **Heapster**: `heapster`

> **Note:** Heapster is deprecated in PKS v1.3, and Kubernetes has retired Heapster. For more information, see the kubernetes-retired/heapster repository in GitHub.

To enable these add-ons, Ops Manager must run scripts after deploying PKS. You must configure Ops Manager to automatically run these post-deploy scripts.

Solution

Perform the following steps to configure Ops Manager to run post-deploy scripts to deploy the missing add-ons to your cluster.

1. Navigate to `https://YOUR-OPS-MANAGER-FQDN/` in a browser to log in to the Ops Manager Installation Dashboard.
2. Click the Ops Manager tile.
3. Select **Director Config**.
4. Select **Enable Post Deploy Scripts**.

> **Note**: This setting enables post-deploy scripts for all tiles in your Ops Manager installation.

5. Click **Save**.
6. Click the **Installation Dashboard** link to return to the Installation Dashboard.
7. Click **Review Pending Changes**. Review the changes that you made. For more information, see Reviewing Pending Product Changes.
8. Click **Apply Changes**.
9. After Ops Manager finishes applying changes, enter `pks delete-cluster` on the command line to delete the cluster. For more information, see Deleting

Clusters.

10. On the command line, enter `pks create-cluster` to recreate the cluster. For more information, see Creating Clusters.

Resurrecting VMs Causes Incorrect Permissions in vSphere HA

Symptoms

Output resulting from the `bosh vms` command alternates between showing that the VMs are `failing` and showing that the VMs are `running`. The operator must run the `bosh vms` command multiple times to see this cycle.

Explanation

The VMs' permissions are altered during the restarting of the VM so operators have to reset permissions every time the VM reboots or is redeployed.

VMs cannot be successfully resurrected if the resurrection state of your VM is set to `off` or if the the vSphere HA restarts the VM before BOSH is aware that the VM is down. For more information about VM resurrection, see Resurrection in the Cloud Foundry BOSH documentation.

Solution

Run the following command on all of your master and worker VMs:

```
bosh -environment BOSH-DIRECTOR-NAME -deployment DEPLOYMENT-NAME ssh INSTANCE-GROUP-NAME -c "sudo /var/vcap/jobs/kube-controller-manager/bin/pre-start; sudo /var/vcap/jobs/kube-apiserver/bin/post-start"
```

Where:

- `BOSH-DIRECTOR-NAME` is your BOSH Director name.
- `DEPLOYMENT-NAME` is the name of your BOSH deployment.
- `INSTANCE-GROUP-NAME` is the name of the BOSH instance group you are referencing.

The above command, when applied to each VM, gives your VMs the correct permissions.

Worker Node Hangs Indefinitely

Symptoms

After making your selection in the **Upgrade all clusters errand** section, the worker node might hang indefinitely. For more information on monitoring the **Upgrade all clusters errand** using the BOSH CLI, see Upgrade the PKS Tile in *Upgrading PKS*.

Explanation

During the PKS tile upgrade process, worker nodes are cordoned and drained. This drain is dependent on Kubernetes being able to unschedule all pods. If Kubernetes is unable to unschedule a pod, then the drain hangs indefinitely. One reason why Kubernetes may be unable to unschedule the node is if the `PodDisruptionBudget` object has been configured in a way that allows 0 disruptions and only a single instance of the pod has been scheduled.

In your spec file, the `.spec.replicas` configuration sets the total amount of replicas that are available in your application. `PodDisruptionBudget` objects can specifies the amount of replicas, proportional to that total, that must be available in your application, regardless of downtime. Operators can configure `PodDisruptionBudget` objects for each application using their spec file.

Some apps deployed using Helm-Charts may have a default `PodDisruptionBudget` set. For more information on configuring `PodDisruptionBudget` objects using a spec file, see Specifying a PodDisruptionBudget in the Kubernetes documentation.

Solution

Configure `.spec.replicas` to be greater than the `PodDisruptionBudget` object.

When the number of replicas configured in `.spec.replicas` is greater than the number of replicas set in the `PodDisruptionBudget` object, disruptions can occur.

For more information, see How Disruption Budgets Work in the Kubernetes documentation. For more information about workload capacity and uptime requirements in PKS, see Prepare to Upgrade in *Upgrading PKS*.

Cannot Authenticate to an OpenID Connect-Enabled Cluster

Symptom

When you authenticate to an OpenID Connect-enabled cluster using an existing kubeconfig file, you see an authentication or authorization error.

Explanation

`users.user.auth-provider.config.id-token` and `users.user.auth-provider.config.refresh-token` contained in the kubeconfig file for the cluster may have expired.

Solution

1. Upgrade the PKS CLI to v1.2.0 or later. To download the PKS CLI, navigate to Pivotal Network. For more information, see Installing the PKS CLI.
2. Obtain a kubeconfig file that contains the new tokens by running the following command:

   ```
   pks get-credentials CLUSTER-NAME
   ```

   Where `CLUSTER-NAME` is the name of your cluster.
3. Connect to the cluster using kubectl.

If you continue to see an authentication or authorization error, verify that you have sufficient access permissions for the cluster.

Error: Failed Jobs

Symptom

In stdout or log files, you see an error message referencing `post-start scripts failed` or `Failed Jobs`.

Explanation

After deploying PKS, Ops Manager runs scripts to start a number of jobs. You must configure Ops Manager to automatically run these post-deploy scripts.

Solution

Perform the following steps to configure Ops Manager to run post-deploy scripts.

1. Navigate to `https://YOUR-OPS-MANAGER-FQDN/` in a browser to log in to the Ops Manager Installation Dashboard.
2. Click the BOSH Director tile.
3. Select **Director Config**.
4. Select **Enable Post Deploy Scripts**.

   Note: This setting enables post-deploy scripts for all tiles in your Ops Manager installation.
5. Click **Save**.
6. Click the **Installation Dashboard** link to return to the Installation Dashboard.
7. Click **Review Pending Changes**. Review the changes that you made. For more information, see Reviewing Pending Product Changes.
8. Click **Apply Changes**.
9. (Optional) If it is a new deployment of PKS, follow the steps below:
   a. On the command line, enter `pks delete-cluster` to delete the cluster. For more information, see Deleting Clusters.
   b. Enter `pks create-cluster` to recreate the cluster. For more information, see Creating Clusters.

Error: No Such Host

Symptom

In stdout or log files, you see an error message that includes `lookup vm-WORKER-NODE-GUID on IP-ADDRESS: no such host`.

Explanation

This error occurs on GCP when the Ops Manager Director tile uses 8.8.8.8 as the DNS server. When this IP range is in use, the master node cannot locate the route to the worker nodes.

Solution

Use the Google internal DNS range, 169.254.169.254, as the DNS server.

Error: FailedMount

Symptom

In Kubernetes log files, you see a `Warning` event from kubelet with `FailedMount` as the reason.

Explanation

A persistent volume fails to connect to the Kubernetes cluster worker VM.

Diagnostics

- In your cloud provider console, verify that volumes are being created and attached to nodes.
- From the Kubernetes cluster master node, check the controller manager logs for errors attaching persistent volumes.
- From the Kubernetes cluster worker node, check kubelet for errors attaching persistent volumes.

Error: Login Failed

Symptom

PKS login command failed with an error “Credentials were rejected, please try again.”

Explanation

You may experience this issue when a large number pods are running continuously in your PKS deployment.

As a result, binary logs that track pod information accumulate over time and fill up the persistent disk of the Pivotal Container Service VM.

Note: Binary logs on the PKS control plane are configured to be purged after a certain number of days.

Solution

1. Check the total number of pods in your PKS deployments.
2. If there are a large number of pods such as over 1,000 pods, then check the amount of available persistent disk space on the Pivotal Container Service VM.
3. If available disk space is low, increase the amount of persistent disk storage on the Pivotal Container Service VM depending on the number of pods in your PKS deployment. Refer to the table in the following section.

Storage Requirements for Large Numbers of Pods

If you expect the cluster workload to run a large number of pods continuously, then increase the size of persistent disk storage allocated to the the Pivotal Container Service VM as follows:

Number of Pods	Storage (Persistent Disk) Requirement ^*
1,000 pods	20 GB
5,000 pods	100 GB
10,000 pods	200 GB

Number of Pods	Storage (Persistent Disk) Requirement ^*
50,000 pods	1,000 GB

Please send any feedback you have to pks-feedback@pivotal.io.

PKS CLI

Page last updated:

This topic describes how to use the Pivotal Container Service Command Line Interface (PKS CLI) to interact with the PKS API.

The PKS CLI is used to create, manage, and delete Kubernetes clusters. To deploy workloads to a Kubernetes cluster created using the PKS CLI, use the Kubernetes CLI, kubectl.

Current Version: 1.3.0-build.125

pks cluster

View the details of the cluster

Synopsis

Run this command to see details of your cluster such as name, host, port, ID, number of worker nodes, last operation, etc.

```
pks cluster [flags]
```

Examples

```
pks cluster my-cluster
```

Options

```
-h, --help   help for cluster
    --json   Return the PKS-API output as json
```

pks clusters

Show all clusters created with PKS

Synopsis

This command describes the clusters created via PKS, and the last action taken on the cluster

```
pks clusters [flags]
```

Examples

```
pks clusters
```

Options

```
-h, --help   help for clusters
    --json   Return the PKS-API output as json
```

pks create-cluster

Creates a kubernetes cluster, requires cluster name, an external host name, and plan

Synopsis

Create-cluster requires a cluster name, as well as an external hostname and plan. External hostname can be a loadbalancer, from which you access your kubernetes API (aka, your cluster control plane)

```
pks create-cluster <cluster-name> [flags]
```

Examples

```
pks create-cluster my-cluster --external-hostname example.hostname --plan production
```

Options

```
-e, --external-hostname string   Address from which to access Kubernetes API
-h, --help                       help for create-cluster
    --json                       Return the PKS-API output as json
    --network-profile string     Optional, network profile name (NSX-T only)
    --non-interactive            Don't ask for user input
-n, --num-nodes string           Number of worker nodes
-p, --plan string                Preconfigured plans. Run pks plans for more details
    --wait                       Wait for the operation to finish
```

pks create-network-profile

Create a network profile

Synopsis

Create network profile requires a path to the profile JSON file (Only applicable for NSX-T)

```
pks create-network-profile <network-profile-JSON-path> [flags]
```

Examples

```
pks create-network-profile my-network-profile.json
```

Options

```
-h, --help   help for create-network-profile
```

pks create-sink

Creates a sink for sending all log data to syslog://

Synopsis

Creates a sink for sending all log data to syslog://

```
pks create-sink <cluster-name> <sink-url> [--name sink-name] [flags]
```

Examples

```
pks create-sink my-cluster syslog://example.com:12345
```

Options

```
-h, --help          help for create-sink
    --name string   Specify a custom name for the sink
```

pks delete-cluster

Deletes a kubernetes cluster, requires cluster name

Synopsis

Delete-cluster requires a cluster name.

```
pks delete-cluster <cluster-name> [flags]
```

Examples

```
pks delete-cluster my-cluster
```

Options

```
-h, --help             help for delete-cluster
    --non-interactive  Don't ask for user input
    --wait             Wait for the operation to finish
```

pks delete-network-profile

Delete a network profile

Synopsis

Deletes network profile. Requires a network profile name (Only applicable for NSX-T). Cannot be deleted if in use

```
pks delete-network-profile PROFILE_NAME [flags]
```

Examples

```
pks delete-network-profile my-network-profile
```

Options

```
-h, --help              help for delete-network-profile
    --non-interactive   Don't ask for user input
```

pks delete-sink

Deletes a sink from the given cluster

Synopsis

Deletes a sink from the given cluster

```
pks delete-sink <cluster-name> [--name sink-name] [flags]
```

Examples

```
pks delete-sink my-cluster
```

Options

```
-h, --help          help for delete-sink
    --name string   Specify a custom name for the sink
```

pks get-credentials

Allows you to connect to a cluster and use kubectl

Synopsis

Run this command in order to update a kubeconfig file so you can access the cluster through kubectl

```
pks get-credentials <cluster-name> [flags]
```

Examples

```
pks get-credentials my-cluster
```

Options

```
-h, --help   help for get-credentials
```

pks login

Log in to PKS

Synopsis

The login command requires -a to target the IP of your PKS API, -u for username and -p for password

```
pks login [flags]
```

Examples

```
pks login -a <API> -u <USERNAME> -p <PASSWORD> [--ca-cert <PATH TO CERT> | -k]

pks login -a <API> --client-name <CLIENT NAME> --client-secret <CLIENT SECRET> [--ca-cert <PATH TO CERT> | -k]
```

Options

```
-a, --api string              The PKS API server URI
    --ca-cert string          Path to CA Cert for PKS API
    --client-name string      Client name
    --client-secret string    Client secret
-h, --help                    help for login
-p, --password string         Password
-k, --skip-ssl-validation     Skip SSL Validation
    --skip-ssl-verification   Skip SSL Verification (DEPRECATED: use --skip-ssl-validation)
-u, --username string         Username
```

pks logout

Log out of PKS

Synopsis

Log out of PKS. Does not remove kubeconfig credentials or kubectl access.

```
pks logout [flags]
```

Examples

```
pks logout
```

Options

```
-h, --help   help for logout
```

pks network-profile

View a network profile

Synopsis

View saved network profile configuration

```
pks network-profile <profile-name> [flags]
```

Examples

```
pks network-profile large-lb-profile
```

Options

```
-h, --help   help for network-profile
    --json   Return the PKS-API output as json
```

pks network-profiles

Show all network profiles created with PKS

Synopsis

Lists and describes network profiles

```
pks network-profiles [flags]
```

Examples

```
pks network-profiles
```

Options

```
-h, --help   help for network-profiles
    --json   Return the PKS-API output as json
```

pks plans

View the preconfigured plans available

Synopsis

This command describes the preconfigured plans available

```
pks plans [flags]
```

Examples

```
pks plans
```

Options

```
-h, --help   help for plans
    --json   Return the PKS-API output as json
```

pks resize

Changes the number of worker nodes for a cluster

Synopsis

Resize requires a cluster name, and the number of desired worker nodes. Users can scale up clusters to the plan defined maximum number of worker nodes, or scale down clusters to one node

```
pks resize <cluster-name> [flags]
```

Examples

```
pks resize my-cluster --num-nodes 5
```

Options

```
-h, --help              help for resize
    --json              Return the PKS-API output as json. Only applicable when used with --wait flag
    --non-interactive   Don't ask for user input
-n, --num-nodes int32   Number of worker nodes (default 1)
    --wait              Wait for the operation to finish
```

pks sinks

List sinks for the given cluster

Synopsis

List sinks for the given cluster

```
pks sinks <cluster-name> [flags]
```

Examples

```
pks sinks
```

Options

```
-h, --help   help for sinks
    --json   Return the PKS-API output as json
```

Please send any feedback you have to pks-feedback@pivotal.io.